Endgeräteunabhängige Realisierung von interaktiven Client/Server-Anwendungen am Beispiel eines integrierten Web- und WAP-Portaldienstes

Diplomarbeit

von

Rudolf Bartel

Technische Universität Hamburg-Harburg
Elektrotechnik/Technische Informatik

Gutachter:
Prof. Dr. Florian Matthes

Zweitgutachter:
Prof. Dr.-Ing. Rolf-Rainer Grigat

Betreuer:
Dipl.-Inform. Holm Wegner

Technische Universität Hamburg-Harburg
Arbeitsbereich Softwaresysteme

Inhaltsverzeichnis

1 EINLEITUNG 1

1.1 Hintergründe und Motivation 1

1.2 Ziel der Arbeit 1

1.3 Gliederung 2

2 ARCHITEKTUR, MODELL UND FUNKTIONSUMFANG EINES PORTALS 3

2.1 Bestehende Internet-Technologien 3

2.2 Architektur eines Web-Portals 6

2.3 Anforderungen an den Funktionsumfang eines Web-Portals 8

2.4 Technische Voraussetzungen für mobile interaktive Dienste 9
2.4.1 Grundkonzept der Mobilfunknetze 9
2.4.2 GSM 10
2.4.3 HSCD 10
2.4.4 GPRS 10
2.4.5 EDGE 10
2.4.6 UMTS 11

2.5 Mobiles Portal 11

2.6 Geräteunabhängigkeit am Beispiel von WAP 11

2.7 Bestehende Lösungen 14
2.7.1 Portal-to-Go 14
2.7.2 Prism von Spyglass 15
2.7.3 TRANSWAP 16

2.8 Architektur des infoAsset Brokers 16
2.8.1 Session-Tracking 18
2.8.2 Substitution 18
2.8.3 Handler 19
2.8.4 Generelle Trennung der projektspezifischen Oberfläche von generischer Software 20
2.8.5 Funktionsumfang des infoAsset Brokers 21

3 WAP UND WML, GRUNDLAGEN DER ARBEIT 22

3.1 WAP-Forum 22

3.2 WAP-Spezifikation 22
3.2.1 WAP-Schichtenmodell 22
3.2.2 WAP-Distributionsmodell 25
3.2.3 WML 27

3.2.4 WML-Elemente 29
3.2.4.1 Variablen 29
3.2.4.2 Links 31
3.2.4.3 Tasks 32
3.2.4.4 Softkeys 32
3.2.4.5 Events 33
3.2.4.6 Templates 34
3.2.4.7 Benutzereingaben 37
3.2.4.8 Textformatierung 39
3.2.4.9 Tabellen 40
3.2.4.10 Bilder 40
3.2.4.11 Mechanismen zum Steuern des internen Zustandes des UA 41
3.2.5 WMLScript 42
3.2.6 WTA/WTAI 44
3.2.7 WML-Erweiterungen 46
3.2.7.1 VCF 46
3.2.7.2 VCS 46
3.2.7.3 Klingeltöne 47

3.3 Push 47

3.4 WAP-Sicherheit 48

3.5 User-Agent-Profile 50

3.6 Diskussion des Standards aus der Sicht des Anwendungsentwicklers 51
3.6.1 Vorteile 51
3.6.2 Nachteile 52
3.6.3 Die Formulierung der WAP-Spezifikation und ihr Einfluss auf die Implementierung 53

3.7 Alternativen zu WAP (Dienste und Technologien) 54
3.7.1 HDML 54
3.7.2 iMode 54
3.7.3 Channels 55
3.7.4 Web Clipping Architecture 55

4 ENDGERÄTEBESCHREIBUNG 56

4.1 Allgemeine Beschreibung der verwendeten Geräte 56

4.2 Übersicht über verwendete Geräte 57
4.2.1 Nokia 7110 57
4.2.2 Nokia 6210 58
4.2.3 Siemens C35i 59
4.2.4 Sony CMD-Z5 60
4.2.5 Palm m100 61

4.3 Unterschiede der WAP-Implementierung der Geräte 62
4.3.1 Titel 62
4.3.2 Softkeys 63
4.3.3 Linklisten 65

4.3.4 Texteingabe (`format`-Parameter) ... 65
4.3.5 Textformatierungen ... 68
4.3.6 Zeilenumbruch ... 68
4.3.7 Auswahllisten ... 69
4.3.8 Tabellen ... 69
4.3.9 Bilder ... 70
4.3.10 Links ... 71
4.3.11 Cache-Verhalten ... 72
4.3.12 Deckgrößenbeschränkung ... 72

4.4 Ergebnisse der Problemanalyse ... 72

5 ENTWURF UND REALISIERUNG EINES GERÄTEUNABHÄNGIGEN WAP-PORTALS ... 74

5.1 Funktionsumfang des WAP-Portals ... 74

5.2 Techniken zur Realisierung der Endgeräteunabhängigkeit ... 76
5.2.1 Portable WAP-Anwendung ... 76
5.2.2 Geräteoptimierte Seiten ... 76
5.2.3 Automatische Generierung ... 78

5.3 Richtlinien für Aufbau von WML- und WMLPLUS-Seiten ... 79
5.3.1 Aufbau der WML- und WMLPLUS-Seiten ... 79
5.3.2 Kartendefinitionen ... 80
5.3.3 Karteninhalt ... 81
5.3.3.1 Sitzungsverwaltung ... 81
5.3.3.2 Links ... 81
5.3.3.3 Texteingabe ... 82
5.3.3.4 Auswahllisten ... 83
5.3.3.5 Bilder ... 83
5.3.4 DoElemente ... 84
5.3.4.1 Rücksprung zur zuletzt angezeigten Karte ... 84
5.3.4.2 "Option"-Softkey (Kartenorientierte Navigationsunterstützung) ... 84
5.3.5 template-Tag ... 85

5.4 Softwaretechnische Umsetzung ... 85
5.4.1 Endgeräteklassifikation ... 86
5.4.2 Automatische Identifikation der Geräte ... 87
5.4.3 Endgeräteverwaltung ... 88
5.4.4 Konfiguration ... 89
5.4.5 Redundanzfreie Speicherung und Suchstrategie ... 91
5.4.6 Generierung von Templates ... 92
5.4.7 Cache-Verhalten und Zähler ... 93
5.4.8 Detektieren der Rückwärtsnavigation nach dem Aufruf einer Seite mit Seiteneffekt ... 94
5.4.9 Deckgrößenbeschränkung und Seitenlistenvorlagen ... 96

5.5 Generalisierung von Geräteunabhängigkeit ... 97

6 ZUSAMMENFASSUNG, BEWERTUNG & AUSBLICK ... 98

ABKÜRZUNGEN 100

GLOSSAR 101

LITERATUR- UND QUELLENVERZEICHNIS 103

1 Einleitung

1.1 Hintergründe und Motivation

> In the beginning, Berners-Lee created HTML and the Web. And the Web was without form, and void; and darkness was upon the face of the deep. And the Spirit of Tim moved upon the face of the routers. And Tim said, "Let there be tags," and there were tags. And Tim said, "Let there be design in the midst of the content, and let it divide the content from the content." And in the beginning there was universality, and everyone could parse the HTML, read the web pages, and Berners-Lee saw everything was good. And the night followed the day, and the years passed …
>
> Josh Smith über das Netz der Netze in „The WAP Vision" Artikel

So oder so ähnlich haben sich die Gründer des WAP-Forums im Jahre 1997 gefühlt, als das WAP-Forum von ihnen ins Leben gerufen wurde. Es waren die vier größten Infrastrukturanbieter für den Mobilfunk: Nokia, Phone.com (damals Unwired Planet), Ericsson und Motorola. Heute repräsentieren die Mitglieder des Forums über 90% aller Mobilfunkanbieter sowie einige Hard- und Softwarehersteller wie Microsoft, HP und Intel. Das Ziel des Forums war es, einen Standard für den Datenverkehr in Mobilfunknetzen zu entwickeln, der sowohl auf bestehenden Strukturen, wie dem Internet, beruht, als auch den Eigenschaften der mobilen Datenkommunikation, wie hohe Latenzzeiten und geringe Bandbreite der Funknetze, schwache Rechenleistung der mobilen Geräte, begrenzte Ein- und Ausgabemöglichkeiten der Benutzerschnittstelle, Rechnung trägt.
In Deutschland wurde WAP spätestens Ende 1999 mit der Einführung der ersten WAP-fähigen mobilen Telefone ein Begriff. WAP wurde als die Technologie der Zukunft und als ein Standard, der es allen Besitzern von WAP-fähigen Geräten ermöglicht, auf speziell aufbereitete Internetinhalte von überall zuzugreifen, angepriesen.
Das WAP-Forum versuchte, die Fehler, die früher gemacht wurden, zu vermeiden. So wurde WML, das mobile Gegenstück zu HTML, als eine XML-Anwendung definiert und nutzt somit die Vorteile von XML im vollen Umfang. Dazu zählen z.B. flexible Erweiterbarkeit durch zusätzliche Elemente, Unterstützung von verschiedenen Zeichencodes und einfache Generierung aus XML mittels XSLT.
Die WAP-Spezifikation, i.e. der WML-Teil der Spezifikation, wurde so formuliert, dass es den Geräteherstellern erlaubte, die WML-Elemente unterschiedlich zu implementieren. Dadurch wurde die Situation geschaffen, die an den „Krieg" zwischen Internet Explorer und Netscape erinnert, mit dem Unterschied, dass es auf dem Markt über 27 WAP-fähige Geräte von 10 Herstellern gibt. Zu Beginn der Arbeit waren es gerade 5 Geräte. Es reicht also nicht, jeweils zwei Versionen zu entwickeln, sondern man muss praktisch für jedes Gerät Optimierung am WML-Code vornehmen. Dadurch entsteht der Bedarf, den Optimierungsaufwand durch geeignete Mechanismen, z.B. durch automatische Generierung von geräteoptimierten Seiten, zu reduzieren.

1.2 Ziel der Arbeit

Vor diesem Hintergrund entstand die Aufgabenstellung dieser Arbeit. Es soll ein bestehendes Web-Portal um einen WAP-Portal erweitert werden. Dabei soll die Ausgabe der WML-Seiten geräteoptimiert erfolgen.
Als Web-Portal, das zu erweitern gilt, kommt der infoAsset Broker der Firma infoAsset zum Einsatz. In dieser Arbeit werden die Anforderungen, die an ein Portal gestellt werden,

dargestellt und spezifiziert. Weiterhin werden die spezifischen Eigenschaften eines mobilen Portals erläutert. Die Beschreibung des WAP-Standards und seiner Implementierungen in verschiedenen Geräten bilden die Grundlage für die WAP-Portal-Erweiterung.

1.3 Gliederung

Kapitel 2 befasst sich mit den technischen Voraussetzungen, die einem Portal zugrunde liegen. Insbesondere werden die im Internet eingesetzten Technologien und das Modell eines interaktiven Web-Portals erläutert. In diesem Kapitel werden auch die Voraussetzungen für ein mobiles Portal, wie Mobilfunknetze, ihre Eigenschaften und Einfluss auf die mobile Datenkommunikation geschildert. Des weiteren wird im Kapitel 2.6 der Begriff der Endgeräteunabhängigkeit erläutert. Es werden auch die Ansätze zur Realisierung der Geräteunabhängigkeit beschrieben. Im Kapitel 2.8 werden der infoAsset Broker, seine Architektur und die Grundfunktionen, wie Session-Tracking und Substitution, behandelt.

Da WAP die Grundlage dieser Arbeit bildet, wird der WAP-Standard beschrieben, und dessen Stärken und Schwächen verdeutlicht. Kapitel 3 beschreibt den WAP-Standard in seinen Einzelheiten: WAP-Modell, WML, WMLScript und die Auswirkung der in WAP-Spezifikation verwendeten Formulierungen. Ein kurzer Ausblick auf die zu WAP vorhandenen Alternativen wird am Ende des Kapitels gemacht.

Im Kapitel 4 werden die Geräte und ihre Besonderheiten in bezug auf WAP-Implementierung, auf die die WML-Seiten optimiert wurden, behandelt. Die Auswirkung der Geräteoptimierung wird beispielhaft an einigen WML-Code-Fragmenten gezeigt.

Kapitel 5 beschäftigt sich mit der Vorgehensweise bei der Realisierung und mit den Technologien, die bei der Lösung eingesetzt wurden. Außerdem wird in diesem Kapitel das Lösungskonzept im Detail aufgezeigt.

Kapitel 6 wird mit der Zusammenfassung der Erkenntnisse sowie Bewertung und Ausblick abschließen.

2 Architektur, Modell und Funktionsumfang eines Portals

2.1 Bestehende Internet-Technologien

Um auf die Architektur eines Web-Portals einzugehen, ist es wichtig einen Überblick über die bestehenden Technologien zu verschaffen. Diese können auf verschiedenen Abstraktionsebenen betrachtet werden. Eine davon ist das Schichtenmodell, das in diesem Kapitel verwendet wird.
Das Internet basiert auf einem Stapel von Protokollen, die die Schichten bilden, so dass die oben angesiedelten Protokolle ein höheres Abstraktionslevel haben als die Schichten darunter liegenden Schichten. Das Internet lässt sich in Anlehnung an das ISO-OSI-Schichtenmodell [KAU-94] beschreiben.
Das OSI-Referenzmodell enthält folgende sieben Schichten:

Abbildung 1: ISO-OSI-Referrenzmodell

Jede Schicht bildet einen Service, der nur von den Nachbarschichten benutzt werden kann. Dabei kann die Implementierung der jeweiligen Schichten sich in Abhängigkeit vom eingesetzten System unterscheiden. Die Schnittstellen zwischen den Schichten sind wohldefiniert und erlauben dadurch den Austausch der Schichten-Implementierungen, ohne Eingriffe in die übrigen Schichten vorzunehmen.
Die sieben Schichten bilden jeweils einen Rahmen für die Entwicklung von Protokollen und Standards.
Die Bitübertragungsschicht ist die unterste Schicht und definiert die Struktur und die Methoden zur Übertragung von einzelnen Bits.
Die Sicherungsschicht arbeitet nicht mit einzelnen Bits, sondern mit Datenpaketen. Sie enthält Fehlererkennungs- und -korrekturmechanismen.
Die Vermittlungsschicht übernimmt die Routing-Aufgaben. Sie realisiert eine Ende-zu-Ende-Verbindung und sorgt dafür, dass die Datenpakete den richtigen Kommunikationspartner beim Transport über ein Netzwerk erreichen.
Die Transportschicht hat auch Ende-zu-Ende-Charakter, kümmert sich aber um die Übertragungsaspekte, so sieht diese Schicht Daten-Quellen und -Senken.

Die Kommunikationssteuerungsschicht dient als Verbindung zwischen zwei Anwendungen oder Prozessen, die miteinander kommunizieren.
Die Darstellungsschicht sorgt für die Transformation der übertragenen Daten, die in einem Zwischenformat vorliegen, in ein auf dem System verwendetes Format.
Die Anwendungsschicht stellt bestimmte Services wie Electronic Mail, File Transfer, etc. dar.

Bei dem OSI-Schichtenmodell handelt es sich um eine Referenz, die sich von den vorhandenen Implementierungen unterscheiden kann. So stimmen nicht alle Schichten im Internet-Schichtenmodell mit denen des OSI-Referenzmodells überein. Insbesondere wird es vom OSI-Referenzmodell abgewichen, in dem der Zugriff nicht nur auf die Nachbarschichten, sondern auch auf die weiter gelegenen möglich ist.
Die oberste Schicht des Internet-Schichtenmodells besteht aus den Komponenten, die die Daten verarbeiten, welche von den darunter liegenden Schichten geliefert werden. Diese Schicht kann in Form eines HTML-Browsers mit oder ohne Skript-, Java-Applet- oder ActiveX-Unterstützung, sowie in Form eines in einer höheren Programmiersprache realisierten Clients vorhanden sein.
HTTP bildet eine Schicht, mit deren Hilfe der Client die HTML-Seiten von dem Server erhält.
Die TLS-SSL Schicht ist die Sicherheitsschicht, nicht zu verwechseln mit der Sicherungsschicht des ISO-Referenzmodells, die zum Aufbau einer sicheren Verbindung mit Hilfe von Verschlüsselungsoperationen verwendet wird. D.h. sie stellt sicher, dass die Unbefugten die Daten weder abhören noch diese unbemerkt manipulieren können. Die Sicherheitsschicht ist optional und wird nur dann eingesetzt, wenn vertrauliche Daten ausgetauscht werden.
Die oben beschriebenen Schichten des Internetschichtenmodells lassen sich den Schichten 5 bis 7 zuordnen, wobei keine klare Eins-Zu-Eins-Zuordnung möglich ist. Da z.B. das OSI-Referenzmodell keine Sicherheitsschicht vorsieht und die oberste Schicht des Internetmodells eigentlich nicht zur Anwendungsschicht gehört.
Die darunter liegende Schicht ist die TCP/IP-Schicht. Sie ist für das sichere Erreichen der Daten zuständig. Diese Internet-Schicht wird im OSI-Referenzmodell durch Schichten 3 und 4 repräsentiert. Dabei entspricht TCP der Transport- und IP der Vermittlungsschicht.
Unter TCP/IP-Schicht liegt das physikalisch verwendete Netz wie Ethernet, Fast Ethernet, Token-Ring. Sie entspricht im OSI-Schichtenmodell der Verbindungssicherungs- und der Bitübertragungsschicht.
Das Bild veranschaulicht das Internet-Schichtenmodell aus Client-Sicht:

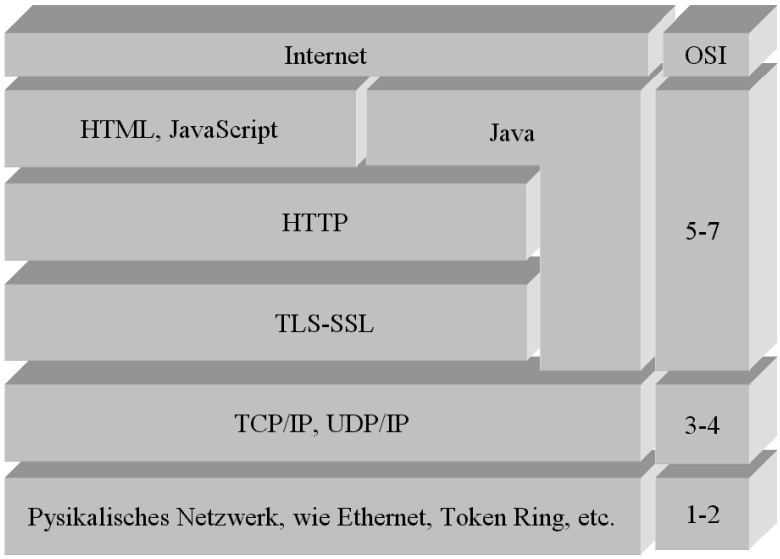


Abbildung 2: Internet-Schichtenmodell

Zur Kommunikation zwischen Client und Server wird bei Web-basierten Anwendungen in der Regel das zustandslose HTTP eingesetzt. HTTP spezifiziert die Datenstrukturen, die sowohl die Nutzdaten als auch zusätzliche Steuerungsinformationen, die Header, enthalten.
Die Header liefern die Informationen über die Fähigkeiten der Kommunikationspartner.
Das Protokoll ist dabei zustandslos, d.h. es baut einen Kanal nur für eine Anfrage auf und schließt die Verbindung nach einer erfolgreichen Übertragung. Die Zustandslosigkeit von HTTP zwingt zum Verwenden zusätzlicher Mechanismen, um die Verbindung über die mehreren Anfragen während einer Sitzung zu erhalten (s. Kapitel 2.8.1).

Als Client einer Web-basierten Anwendung dient normalerweise ein HTML-Browser. Moderne Browser unterstützen außerdem eine oder mehrere Skriptsprachen, wie JavaScript und VBScript. Durch den Einsatz von Skriptsprachen ist eine clientseitige Vorverarbeitung der Daten, wie Benutzereingaben, Verwaltung und Dynamisierung der Bildschirminhalte, Verwaltung des Client-Zustandes und Senden der Zustandsänderung an den Server, möglich. Außerdem können die Browser Java-Applets oder ActiveX-Steuerelemente ausführen, die sowohl die Datenverarbeitung als auch die Darstellung von Inhalten übernehmen können. Durch diese Eigenschaften ist es möglich, Web-Anwendungen zu entwickeln, die in der Qualität und dem Funktionsumfang die lokalen Anwendungen erreichen.

Das Bild zeigt die Technologien, die auf HTTP bzw. TCP/IP client- und serverseitig aufsetzen:

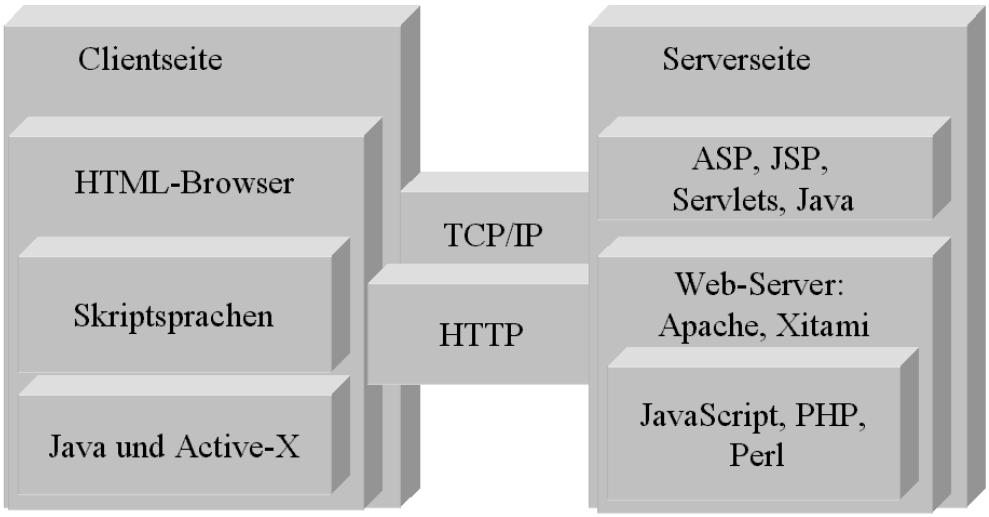


Abbildung 3: Client- und Server-Technologien

Auf der Server-Seite kann eine ganze Palette von Technologien eingesetzt werden, um eine geeignete Umgebung zu schaffen. Einige Technologien wie z.B. ASP (Active Server Pages) sind zum Teil plattformabhängig, dagegen können andere Technologien, wie JSP, Servlets, Java, PHP, Perl, etc., für alle gängigen Systeme verfügbar sein. Die vorhandenen Server wie Apache erlauben das Einbinden von verschiedenen Programmiersprachen und sind sowohl auf Unix-Systemen als auch auf Windows-basierten Systemen vorhanden.
Falls man die volle Funktionalität eines Web-Servers nicht braucht, weil z.B. die zu entwickelnde Web-Anwendung nur wenige HTTP-Methoden, wie get und post nutzt, kann der Web-Server in die Anwendung integriert und in derselben Programmiersprache wie die Anwendung selbst entwickelt werden.

2.2 Architektur eines Web-Portals

In diesem Kapitel wird das Modell eines interaktiven Dienstes beschrieben, der in Form eines Portals realisiert wird, also einer Web-Anwendung, die verschiedene Web-Dienste unter einer Oberfläche vereint.

Das Modell eines interaktiven Dienstes baut auf bestehenden Technologien wie Internet, Client/Server- und Mehrschicht-Architektur auf. Eine Portal-Anwendung kann zudem ein Bestandteil eines größeren verteilten Systems sein, in dem Datenbanken, Kontenmanagementsysteme, Shopping und Billing-Systeme, Unified Messaging Systeme integriert sind.
Mit steigender Komplexität der Anwendung ist es wichtig, diese in logische Bestandteile zu gliedern und zu beschreiben. Genauso, wie das Internet sich in einem Schichtenmodell beschreiben lässt, kann auch die Architektur einer Web-basierte Anwendung als Schichtenmodell dargestellt werden. Dabei hängt die Anzahl der Schichten davon ab, wie komplex die Anwendung ist. Durch die Verwendung der n-tier-Architektur kann man ohne zusätzlichen Mehraufwand die jeweilige Schicht der Anwendung anpassen, um z.B. die Benutzerschnittstelle zu gestalten, ohne die darunter liegende Anwendungslogik zu verändern, oder die verschiedenen Speicherungsmöglichkeiten, z.B. wie Filesystem, Datenbank oder Content Management Systeme zu berücksichtigen. Die vordefinierten Dienste, die in den mittleren Schichten implementiert sind, sollten eine Reihe von Grundfunktionen bieten. Diese Dienste sollen erweiterbar sein, ohne die vorhandenen Funktionen zu ändern.

Die Abbildung 4 gibt eine Aufteilung eines möglichen Web-basierten Dienstes in Schichten an:

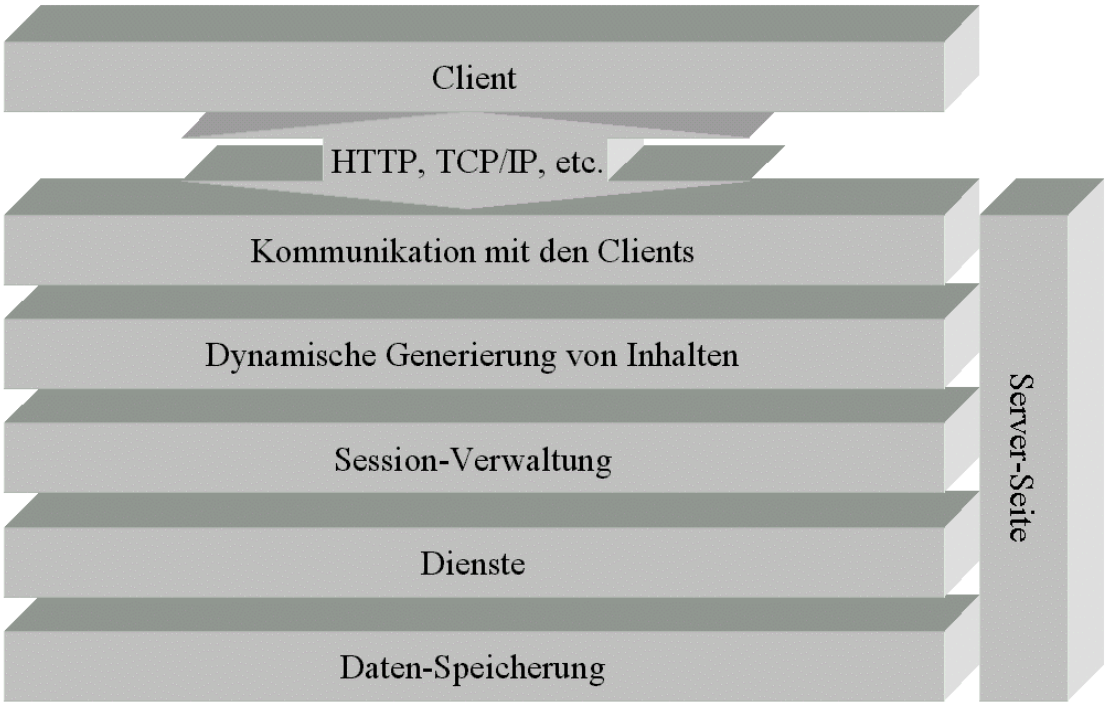


Abbildung 4: Architektur eines Web-Portals

Der Client übernimmt die Rolle der Benutzerschnittstelle und kann die Daten vorverarbeiten, bevor der Datenaustausch mit dem Server stattfindet. Außerdem muss die Konsistenz der Daten und des Client-Zustandes mit dem Server-Zustand gewährleistet sein.
Der Datenaustausch zwischen Client und Server kann mit Hilfe von HTTP oder direkt über TCP/IP erfolgen. Diese Protokolle sind statuslos und müssen, um Session-Tracking zu realisieren, durch geeignete Mechanismen wie zusätzliche HTTP-Parameter oder Cookies erweitert werden. Diese Erweiterung kann in der Schicht der Session-Verwaltung implementiert werden.
Auf der Server-Seite muss eine für die Kommunikation mit dem Client zuständige Schicht vorhanden sein. Die Aufgabe dieser Schicht ist es, z.B. die HTTP-Anfragen entgegen zu nehmen und die Antworten an den Client zu senden. Dabei kann sie die Extraktion der Parameter aus der HTTP-Anfragen oder der User-Agent-Informationen aus den HTTP-Header vornehmen. Diese Schicht kann mit Hilfe eines separaten Web-Servers, der als Vermittler zwischen dem Client und dem Applikationsserver dient, oder aber auch als integraler Bestandteil des Applikationsservers realisiert werden.
Die nächste Schicht ist für die Anwendungslogik und dynamische Generierung von Inhalten zuständig. Die dynamische Generierung von Inhalten kann z.B. als XSLT oder auch als Substitution von Platzhaltern in vorgenerierten Seiten implementiert werden. In einer Web-basierten Anwendung wird die Anwendungslogik auf der Cleint-Seite durch Links realisiert. Die Links enthalten Parameter, die in dieser Schicht ausgewertet werden, wodurch verschiedene Aktionen ausgeführt werden können. Als Reaktion auf Anfragen, die als Parameter die internen Zustände des Clients enthalten, kann eine serverseitige Verarbeitung stattfinden und eine Seite mit bestimmten dynamisch generierten Inhalten an den Client gesendet werden.
Die Session-Verwaltung ist die Schicht, die für die Authentifizierung der Benutzer gegenüber dem System, Identifizierung der Clients während einer Sitzung, Verwaltung und Synchronisation der Clientzustände sowie Feststellen und Verwalten des Client-Typs, zuständig ist.
Die Authentifizierung des Benutzers kann durch die Anmeldung im System mittels Eingabe des Passwortes und des Benutzernamen oder auch durch andere Mechanismen, wie Fingerabdruck-Scan, Chipkarte, etc erfolgen.
Die Identifizierung der Clients wird für die Dauer einer Sitzung bei jeder neuen Anfrage, auch als Session-Tracking bezeichnet, dafür eingesetzt, um die Zuordnung von verschiedenen Clients, die gleichzeitig mit dem System kommunizieren, zu ermöglichen.
Die serverseitige Verwaltung von den Clientzuständen ist deswegen notwendig, weil die Benutzerschnittstelle Elemente enthält, wie Radio-Buttons, Auswahlboxen, etc., mit denen das Verhalten des Systems beeinflusst werden kann. Deswegen besteht der Bedarf, die Einstellungen, die vom Benutzer durchgeführt wurden, an den Server zu übermitteln. Dabei kann die Synchronisation unmittelbar nach der Änderung des Clientzustandes oder erst bei der nächsten Anfrage durchgeführt werden.
Das Detektieren und Verwalten des Client-Types ist für die Implementierung der Geräteunabhängigkeit von Bedeutung. Um z.B. eine an ein bestimmtes Gerät optimierte Seite zu senden, muss das Gerät erkannt und einer Geräteklasse zugeordnet werden.
Die darunter liegende Schicht soll alle Dienste bereitstellen, die den Funktionsumfang eines Portals darstellen. Diese Dienste müssen möglichst modular aufgebaut sein und über eine standardisierte Schnittstelle verfügen. Sie können die Operationen auf Benutzerdaten, wie neuen Benutzer Anlegen, Benutzer Löschen, oder auch z.B. SMS-Senden, Dokument Löschen etc., bereitstellen.
Um die Daten persistent zu speichern, ist eine weitere Schicht vorhanden. Nach oben hin bietet diese Schicht eine Schnittstelle, die von der eigentlichen Speicherungsmethode

abstrahiert. Das hat den Vorteil, dass das System sowohl mit dem Dateien und Datenbanken als auch mit Kontentmanagement-Systemen arbeiten kann.

2.3 Anforderungen an den Funktionsumfang eines Web-Portals

Die Anforderungen an den Umfang der Grundfunktionalitäten sind folgende:

1) Session-Tracking

   Als Session-Tracking wird die Zuordnung eines Clients während einer Sitzung, die auf einem zustandslosen Protokolls wie HTTP aufsetzt, zu den serverseitig ausgeführten Tasks bezeichnet.
   Es gibt eine Reihe von Möglichkeiten während einer Sitzung sicherzustellen, dass der Datenaustausch mit einem und demselben Client erfolgt. Generell wird eine Identifizierung des Clients während einer Sitzung durch den Server benötigt. Es wird eine vom Server generierte Zahl oder ein String an den Client übergeben. Diese Identifizierung wird bei jeder Anfrage vom Clients an den Server als Parameter übergeben, so dass der Client vom Server anhand dieser Kennung eindeutig identifiziert werden kann.
   Eine andere Möglichkeit, Session-Tracking zu realisieren, ist die Verwendung von Cookies. Dabei werden die zur Identifizierung benötigten Daten clientseitig sogar über die Dauer einer Sitzung hinaus gespeichert. Dadurch können Cookies auch für die automatische Anmeldung am System verwendet werden. Der Nachteil von diesem Verfahren ist, dass die Cookies sich vom Benutzer deaktivieren lassen, d.h. die Funktion des Session-Tracking ist nicht mehr gewährleistet.
   Bei Anwendungen, die mobile Telefone als Clients verwenden, kann die Erkennung der Clients anhand der im Gerät oder auf SIM-Karte gespeicherten Identifikationsnummer erfolgen. Den Zugang zu dieser Nummer haben aber nur die Funknetzbetreiber, so dass für die Dritt-Dienstanbieter diese Information nicht zur Verfügung steht.

2) Authentifizierung

   Es soll vom System sichergestellt werden, dass es sich bei der Person, die mit dem System arbeitet, um die dazu berechtigte Person handelt. Zu diesen Zwecken können passwort-basierte Anmeldeverfahren und persistente serverseitige Speicherung von Nutzerdaten eingesetzt werden. Außer den passwort-basierten Verfahren können andere, wie verschiedene biometrische Verfahren, die jedoch zusätzliche Hardware-Anforderungen stellen, eingesetzt werden.

3) Personalisierung

   Der Benutzer soll seine eigenen Einstellungen und Eintragungen durchführen und speichern können. Es soll für den Benutzer die Möglichkeit bestehen, die von ihm benötigten Portal-Dienste aktivieren und deaktivieren, seine Links zu verwalten, Inhalte zu abonnieren oder auch den mobilen Zugang komfortabel über den Web-Interface zu konfigurieren.
   Zur Personalisierung gehört auch die Veraltung von Benutzern, die im System registriert sind. Außerdem soll jeder Benutzer seine eigenen Anmeldeinformationen editieren können. Die Gruppenverwaltung soll ermöglichen, die Benutzer

verschiedenen Gruppen zuzuweisen, und so eine Autorisierung auf Gruppenebene zu bekommen, bestimmte Dienste zu nutzen.
Die Verwaltung von Inhalten, die von einem Content Management System zur Verfügung gestellt werden, gehört auch zum Funktionsumfang eines Portals. Dabei soll es den autorisierten Nutzern ermöglicht werden, die Inhalte zu bearbeiten, freizugeben und zu sperren.

4) Konfigurierbarkeit

Das Portal soll ohne große Änderungen am Kernsystem an die kundenspezifische Projektanforderungen angepasst werden. Dazu gehört sowohl die Anpassung und Erweiterung des Funktionsumfangs als auch die Anpassung der Benutzerschnittstelle, also Design der grafischen Oberflache, sowie der Anmelderestriktionen z.B. in Bezug auf Passwortlänge und Klein-/Großschreibung von Anmeldenahmen.

2.4 Technische Voraussetzungen für mobile interaktive Dienste

Die Entwicklung der Kommunikationsnetze befindet sich heute im Umbruch. Es werden immer wieder neue Übertragungsverfahren entwickelt und eingesetzt. Diese Verfahren bauen teilweise auf bestehenden Technologien auf, manche von denen erfordern jedoch eine vollständig neue Infrastruktur.
Dieses Kapitel befasst sich mit den technischen Hintergründen, die bei der Entwicklung von Standards für mobile Kommunikation, wie WAP, berücksichtigt werden müssen. Hier werden auch die neuen leistungsfähigeren Standards vorgestellt.

2.4.1 Grundkonzept der Mobilfunknetze

Das Grundkonzept der mobilen Kommunikationsnetze [Risch2000] beruht auf Zellen-Metapher. Das ganze Gebiet, das abgedeckt werden soll, wird in Zellen aufgeteilt. Die Ecken der Zellen bilden die Sendestationen. Befindet sich ein Teilnehmer innerhalb einer Zelle, so kann er von mehreren Sendestationen gleichzeitig erreicht werden. Um die Arbeit von Zellen zu koordinieren, sind die Sendestationen mit einer Mobilvermittlungsstelle (MSC - Mobile Switching Center) verbunden. Die Größe der Zellen ist für die Kapazität des Gesamtnetzes ausschlaggebend. D.h. je kleiner die Zellen sind, desto mehr Teilnehmer kann das Netz insgesamt aufnehmen. Das liegt an der begrenzten Anzahl der zur Verfügung stehenden Funkkanäle. Deswegen versucht man, die Größe der Zellen klein zu halten, um mit einer geringerer Leistung senden zu können. Dies bedeutet, dass die Kanäle sich bei einer geringeren Entfernung wiederverwenden lassen, was effektiv zu einem Kapazitätsgewinn führt. Der andere Vorteil liegt im geringeren Energieverbrauch des Mobilfunkgerätes und besseren Signalqualität durch gleichmäßigere Ausleuchtung der Flächen.
Die Mobilfunkvermittlungsstelle ist dafür zuständig, die Informationen aller in der zuständigen Zelle aktiven Benutzer zu verwalten und zu bestimmen, welche der Stationen mit dem Benutzer kommuniziert. Die Vermittlungsstelle ist auch dafür zuständig, bei dem Zellenwechsel des Benutzers die Kontrolle an eine andere Stelle zu übergeben.
Alle digitalen Mobilfunknetze sind paketvermittelnd und eignen sich sowohl für die Sprach- als auch Datenübertragung. Die Hauptcharakteristika der Netze sind die Latenzzeit und Übertragungsrate. Die Latenzzeit drückt die Zeit aus, die seit der Anfrage bis zum Empfang des ersten Bits vergeht, also wie lange es dauert bis die Daten empfangen werden können. Die Latenzzeit in Mobilfunknetzen ist typischer Weise höher als die Latenzzeit der Festnetze. Die

Übertragungsrate gibt an, wie schnell die Daten bei einer bestehenden Verbindung übertragen werden.
Weitere Eigenschaften der Netze sind z.B. die Verbindungsaufbauzeit, d.h. die Zeit, die für die Einwahl in das Netz benötigt wird, oder auch die Verbindungsstabilität, die bei den Mobilfunknetzen generell niedriger als bei den Festnetzen ist, da das Signal durch unvorhersehbare Ursachen wie atmosphärische Störungen, geographische und architektonische Strukturen etc. sehr stark gedämpft werden kann.

2.4.2 GSM

In Europa werden GSM-Mobilfunknetze verwendet. Die Übertragungsrate ist auf 9,6 KBit/s begrenzt. Die Einwahl bei einem Provider oder WAP-Gateway dauert über eine Minute. Die Benutzung von Internetdiensten, die für das Internet konzipiert wurden, ist nicht möglich, da die Inhalte auf Datenraten ab 40 KBit/s ausgelegt sind. Die WAP-Anwendungen können dagegen mit einer Datenrate von 9,6 KBit/s gut benutzt werden. Der große Nachteil, der beim Verwenden von WAP-Anwendungen ist, ist die lange Einwahlzeit, die sich auch beim Wiedereinwählen bemerkbar macht. Das Wiederein wählen ist notwendig, falls die Verbindung länger nicht aktiv benutzt und ein Time-Out-Wert überschritten wird. Ein weiterer Nachteil der GSM-Netze ist die Abrechnung nach der Verbindungsdauer und nicht nach dem Datenvolumen.

2.4.3 HSCD

Die nächste Ausbaustufe, die auf GSM-Netzen basiert, ist HSCD und erlaubt die Transferraten von bis zu 76,8 KBit/s [c´t2000]. HSCD ist seit einem Jahr im Einsatz, allerdings wird die maximale Datenrate auf 28,8 KBit/s begrenzt. HSCD arbeitet mit Bündelung von 2 bis 8 Kanälen und einer Bandbreite von 14,4 KBit/s pro Kanal. Die höhere Datenrate gegenüber einem üblichen GSM-Kanal ist aufgrund der verminderten Fehlerkorrektur möglich. HSCD hat aber folgenden Nachteil von GSM vererbt: um eine Verbindung zu etablieren, muss man sich Einwählen, dadurch bleibt man während der ganzen Sitzung online und blockiert so die knappen Netzressourcen. Außerdem erfolgt die Abrechnung nach Online-Zeit.

2.4.4 GPRS

Ein weiteres Verfahren namens GPRS, das auch auf GSM-Netzen basiert, hat eine noch höhere Übertragungsrate von bis zu 160 KBit/s, erfordert aber weitgehende Eingriffe in die GSM-Infrastruktur. Das Verfahren arbeitet vollständig paketorientiert, d.h. das Endgerät bleibt die ganze Zeit aktiv und belegt die Funkkanäle nur dann, wenn auch tatsächlich die Daten übertragen werden. Die Abrechnung erfolgt nach dem tatsächlich übertragenen Datenvolumen.

2.4.5 EDGE

Mit EDGE werden noch höhere Datenraten von bis zu 473,6 KBit/s ermöglicht. Dies wird durch das Einsetzen neuer Modulationsverfahren erreicht, was einen theoretisch dreifachen Kapazitätsgewinn gegenüber den konventionellen Verfahren mit sich bringt. EDGE erlaubt

adaptives Umschalten zwischen der neuen und alten Modulationsart, da das neue schnellere Verfahren störungsanfälliger ist.

2.4.6 UMTS

Für UMTS wird eine vollständig neue Infrastruktur benötigt. Diese Technologie wird die Übertragungsraten von bis zu 2000 KBit/s für sich nicht bewegende und 384 KBit/s für sich bewegende Teilnehmer bieten.

2.5 Mobiles Portal

In diesem Kapitel werden die Anforderungen an die mobilen interaktiven Dienste, sowie beschränkenden Faktoren und Benutzbarkeitsaspekte beschrieben.

Die mobilen interaktiven Dienste sind von den vorhandenen Mobilfunknetzen, Endgeräten und von der Situation, in der diese benutzt werden, abhängig. Die heute üblichen Netze haben eine typische Bandbreite von 9,6 KBit/Sekunde und eine große Latenzzeit. Die mobilen Geräte haben eine sehr geringe Rechenleistung, minimale Speicherausstattung, einen numerischen Tastatur und ein kleines Display mit einer geringen Farbtiefe. Diese Beschränkungen setzen den Rahmen für den Entwurf eines Standards für mobile Kommunikation wie WAP, aber auch für die Dienstentwickler. Darüber hinaus liegt es in der Verantwortung des Anwendungsentwicklers, welche Inhalte übertragen und dargestellt werden.
Die Inhalte müssen an die Situation, in der sie möglicherweise verwendet werden, angepasst sein. Der mobile Anwender kann einen Dienst beim Gehen oder während einer Konferenzpause, um z.B. seinen Flug zu verlegen, nutzen. D.h. die Information muss sofort ohne langes Suchen und Warten abgerufen werden können, sie muss klar und unmissverständlich sein. Es gilt also, grundsätzliche Überlegungen über den Funktionsumfang und die Inhalte zu machen, die von einem mobilen Portal zur Verfügung gestellt werden.
Die geringe Bandbreite der Verbindung beschränkt den Einsatz von Multimediakomponenten auf einfache meist Schwarz-Weiß-Bilder. Die geringe Größe der Displays erschwert das Lesen und die numerische Tastatur der Mobiltelefone das Eingeben von längeren Texten. Vor diesem Hintergrund ist ein mobiles Portal nur in der Verbindung mit einem Web-Portal sinnvoll, vor allem wenn es sich um einen personalisierbaren Dienst handelt, bei dem umfangreiche Einstellungen und Eingaben nötig sind. So kann man komfortabel längere Eingaben und Konfigurationseinstellungen mit Hilfe des Web-Clients durchführen und kurze Informationspakete mit Hilfe eines mobilen Gerätes abrufen, falls man diese unterwegs benötigt.
Grundsätzlich kann jede Funktion, die in einem Web-Portal vorhanden ist, auch in einem mobilen Portal implementiert werden. Allerdings wird der Informationsgehalt einer mobilen Anwendung durch die oben beschriebenen Eigenschaften stark reduziert.
Es gibt aber auch solche Funktionen, wie Volltextbearbeitung oder andere eingabelastige Funktionen, die auf bestimmten Typen von Geräten (Telefon mit numerischer Tastatur) nur mit einem großen Aufwand vom Endbenutzer, jedoch auf PDA komfortabel ausgeführt werden können.

2.6 Geräteunabhängigkeit am Beispiel von WAP

Die Endgeräteunabhängigkeit beschreibt die Fähigkeit eines Systems, hier eines Servers, eine Vielzahl von verschiedenen Clients optimal zu bedienen, die alle denselben Standard, z.B.

HTML, unterstützen. Eine Erweiterung des Begriffs der Geräteunabhängigkeit führt zum Begriff Medienunabhängigkeit. Die Medienunabhängigkeit ist die Fähigkeit eines Systems, Clients zu bedienen, die verschiedene Formate unterstützen.

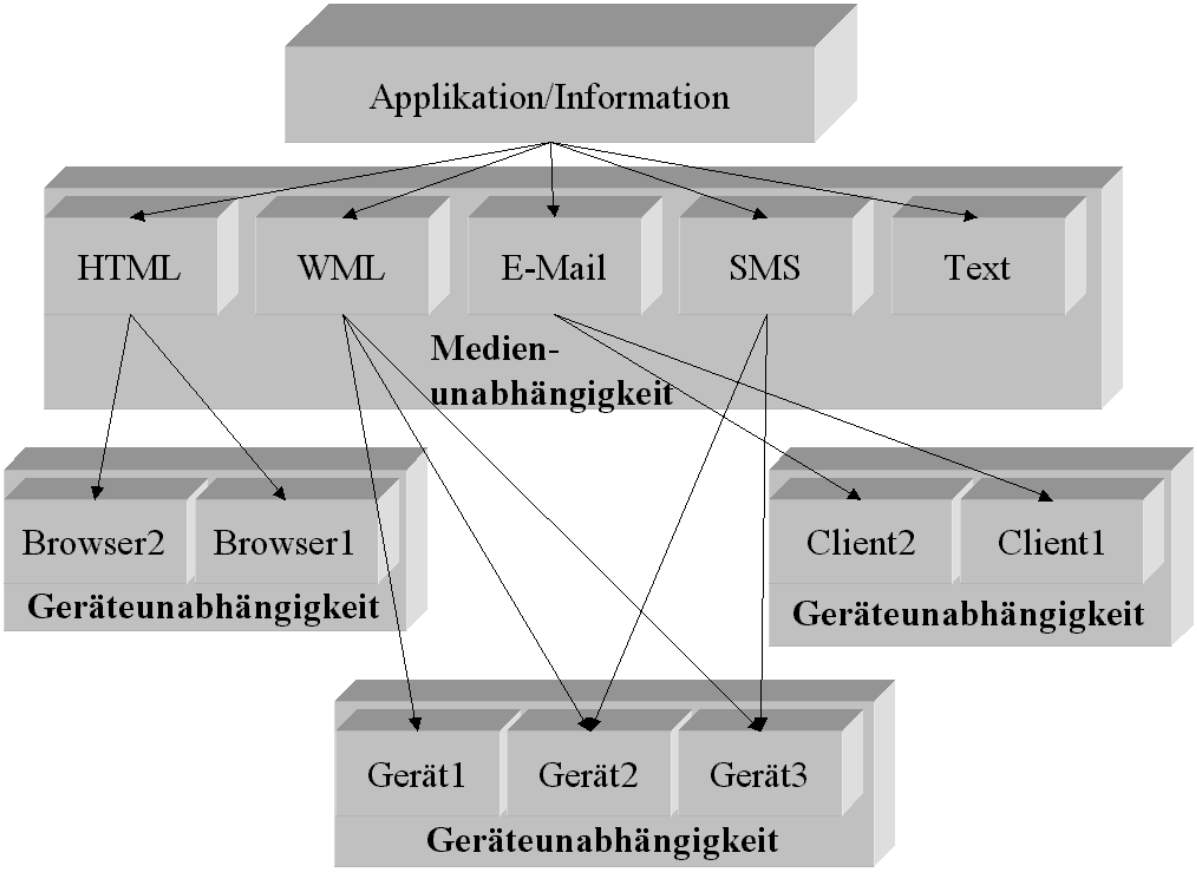


Abbildung 5: Geräteunabhängigkeit vs. Medienunabhängigkeit

Der Begriff Endgeräteunabhängigkeit hat zwei Aspekte:
Endgerät und Unabhängigkeit. Ein Endgerät kann in diesem Fall sowohl ein physikalisches Gerät (Mobiles Telefon, PDA, Kühlschrank mit Web-Anschluss) als auch ein Programm (WAP-Emulator, HTML-Browser, Email-Client, etc) sein. Unter Unabhängigkeit versteht man eine optimale Darstellung der Information auf verschiedenen Geräten. Da Spezifikation eines Zielformats (Word, WML, HTML, ASCII, SMS) die Eigenschaften des Endgerätes nicht berücksichtigen kann, werden zusätzliche Mechanismen benötigt, die diese Funktion erfüllen.
Es stellt sich die Frage, warum man auf die Geräteunabhängigkeit überhaupt eingehen muss, da der WAP-Standard eindeutig beschreiben soll, wie die WML-Tags interpretiert werden sollen. Das grundlegende Problem ist aus der Erfahrung mit den HTML-Browser bekannt: unterschiedliche HTML-Browser stellen den gleichen HTML-Code verschieden dar und interpretieren die Skripte auf verschiedene Weise, so dass verschiedene Browser gesondert behandelt werden müssen.
Die WAP-Spezifikation wurde bewusst vom WAP-Forum so formuliert, dass die Gerätehersteller nicht alle Elemente zu implementieren brauchen. Es wird später im Kapitel 3.6.3 anhand der WAP-Spezifikation gezeigt, welche Formulierungen den Geräteherstellern die Freiheit geben, die gleichen WML-Elemente auf verschiedene Art und Weise zu implementieren. Ein weiterer Aspekt, der eng mit der Geräteunabhängigkeit zusammenhängt, ist die Benutzbarkeit. Der Begriff Benutzbarkeit, oder Usability, umfasst alles, was sich auf

die Optimierung des Quellcodes in Bezug auf Ergonomie, Lesbarkeit, Minimierung der Anzahl der für eine Eingabe oder Navigation erforderlichen Tastenbetätigungen bezieht, also mit der Anpassung des Quellcodes an die spezielle Benutzerschnittstelle eines Endgerätes zu tun hat.
Da viele Geräte Zusatzfunktionen bieten, die im WAP-Standard nicht vorkommen, muss dies auch bei der Implementierung der Geräteunabhängigkeit berücksichtigt werden.
Es besteht also der Bedarf, auf einer Abstraktionsebene alle WAP-fähigen Geräte uniform zu behandeln, jedoch ihre Eigenschaften zu berücksichtigen. Es gibt drei Ansätze, um die Geräteunabhängigkeit zu erreichen:

1. Portablen Code zu schreiben, um Problemelemente oder Problemkombinationen, die bei einigen Endgeräten auftreten, zu vermeiden.
2. Für jedes einzelne Gerät speziell angepassten Code zu schreiben.
3. Den optimierten Code aus einem geräteneutralen Format zu generieren.

Diese Methoden bringen unterschiedlichen Entwicklungsaufwand mit sich. Hier ist eine Abschätzung des Speicherungs- und Implementierungsaufwandes:

- Die erste Möglichkeit hat den Vorteil, dass der Aufwand, die Dateien sowohl zu generieren als auch zu ändern, sehr gering ist. Besteht der Dienst aus N Dateien, so müssen N Dateien gespeichert und im Falle einer Änderung maximal N Mal angefasst werden. Allerdings ist die Optimierung der Benutzbarkeit und Ausnutzung der Zusatzfunktionen, wie z.B. der Zugriff auf Dateien im VCF-Format, nicht möglich.
- Die zweite Möglichkeit bietet eine optimale Anpassung der Dateien an die Endgeräte. Bei einem aus N Dateien bestehenden Dienst und dem Einsatz von M Endgeräten entsteht ein Datenbestand von N*M Dateien, d.h. bei einer Änderung müssen maximal N*M Dateien mindestens aber M Dateien angefasst werden. Darüber hinaus müssen bei jedem neu hinzukommenden Gerät N Dateien neu generiert bzw. angepasst werden.
- Bei der dritten Möglichkeit ist der Speicherungsaufwand genauso groß wie bei der zweiten, jedoch müssen nur N Dateien entwickelt werden, die den Dienst in einer geräteneutralen Form beschreiben, und weitere M Dateien, die die Information über die Geräteeigenschaften enthalten und zum Transformieren der geräteneutralen in die gerätespezifischen Dateien verwendet werden. Daraus ergibt sich der Entwicklungs- und Pflegeaufwand von N+M und liegt somit deutlich niedriger als bei der zweiten Alternative.

Die Abbildung 6 verdeutlicht die drei Möglichkeiten, die Geräteunabhängigkeit zu erreichen.

Da davon ausgegangen werden kann, dass die Geräte in Klassen aufgeteilt werden können und einige Dateien verschiedener Geräte gleich sein können, kann man diese Redundanzen eliminieren und somit den Speicherungsaufwand vermindern. Die Aufteilung der Geräte in Klassen ist deswegen möglich, weil verschiedene Geräte vom gleichen Hersteller beinahe identische Software benutzen und ähnliche Bedienungselemente haben. Es gibt auch Browser-Hersteller, deren Browser von verschiedenen Geräteherstellern eingesetzt werden, so dass verschiedene Geräte ähnliches Verhalten aufweisen können, obwohl sie von verschiedenen Herstellern kommen.

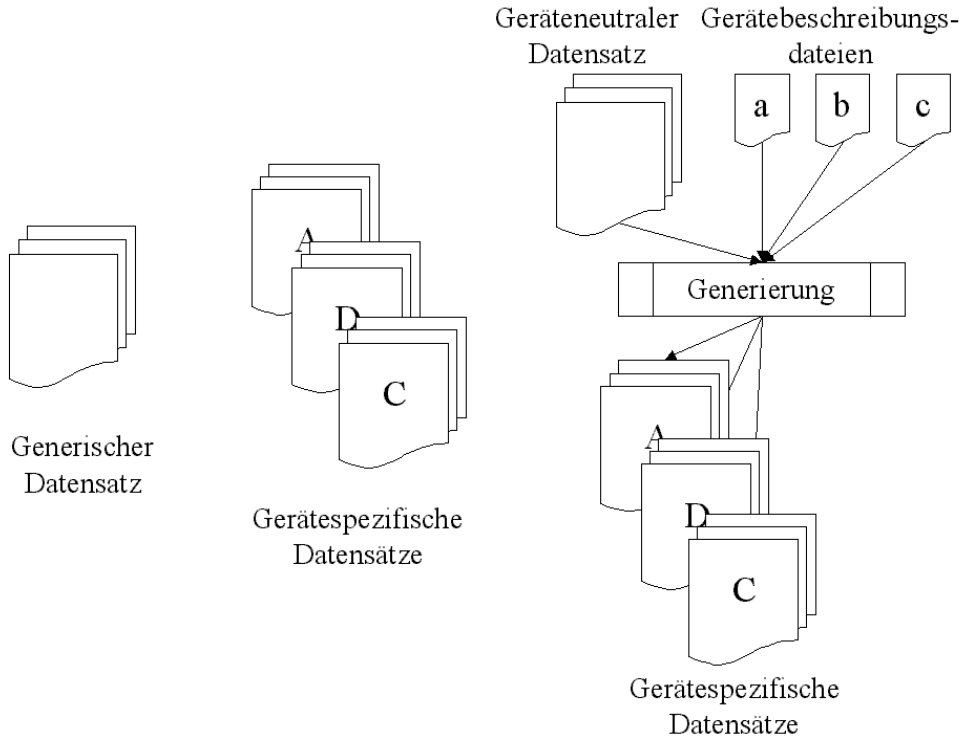


Abbildung 6: drei Implementierungsansätze für die Geräteunabhängigkeit

2.7 Bestehende Lösungen

2.7.1 Portal-to-Go

Oracle bietet eine Lösung für mobile Portal-Dienste, die auf XML basiert. Das Portal setzt auf der Oracle8i Datenbank und dem Oracle Application Server auf. Die Dienste werden in XML definiert, die durch eine DTD validiert werden. Dadurch ist es möglich, die mit Portal-to-Go entwickelten Dienste für eine Reihe von Markup-Sprachen zur Verfügung zu stellen. Die von Portal-to-Go unterstützten Sprachen sind HTML, WML, HTML für PDAs, VoxML.
Diese Lösung verfolgt den medienunabhängigen Ansatz und kann eingesetzt werden, um die Geräteunabhängigkeit zu erreichen. Durch die Verwendung der so genannten Device-Transformer ist es möglich, die Palette von unterstützten Sprachen und Endgeräten ohne Änderung am Dienst zu erweitern [SCNEd2000 und Areh2000]. Die Dienstdefinition kann mittels HTML oder in einem anderen Format erfolgen. Dies wird durch den Einsatz von Adaptern erreicht, die die verschiedenen Quellformate in das Zwischenformat übersetzen. Die Device-Transformer können als Ausgabefilter und die Adapter als Eingabefilter angesehen werden. Das Zwischenformat ist eine XML-Anwendung, die sich eines wohldefinierten Satzes von XML-Elementen bedient, die in einer DTD definiert werden.
Daraus resultieren sowohl Vor- als auch Nachteile. Die Vorteile sind:

- Die Dienste können in einer beliebigen Markup-Sprache, für die es einen Adapter gibt, entwickelt, d.h. der Entwickler muss die Zielsprachen nicht kennen.
- Die syntaktische Korrektheit der entwickelten Seite ist durch die DTD-Validierung sichergestellt.
- Entwicklung von den Transformern, XSL-Dateien, ist einfach, da die Elemente des Zwischenformats vordefiniert sind.

Die Nachteile sind:

- Optimierung und Anpassung an die Eigenarten des Zielformats oder des Zielgerätes sind nur innerhalb der vom Portal-to-Go vorgegebenen Rahmen möglich.
- Die Flexibilität geht verloren, da man starren Strukturen des Zwischenformates folgen muss.
- Erweiterung des Zwischenformats mit eigenen Elementen ist nicht möglich.

Die Abbildung zeigt die Portal-to-Go Architektur:

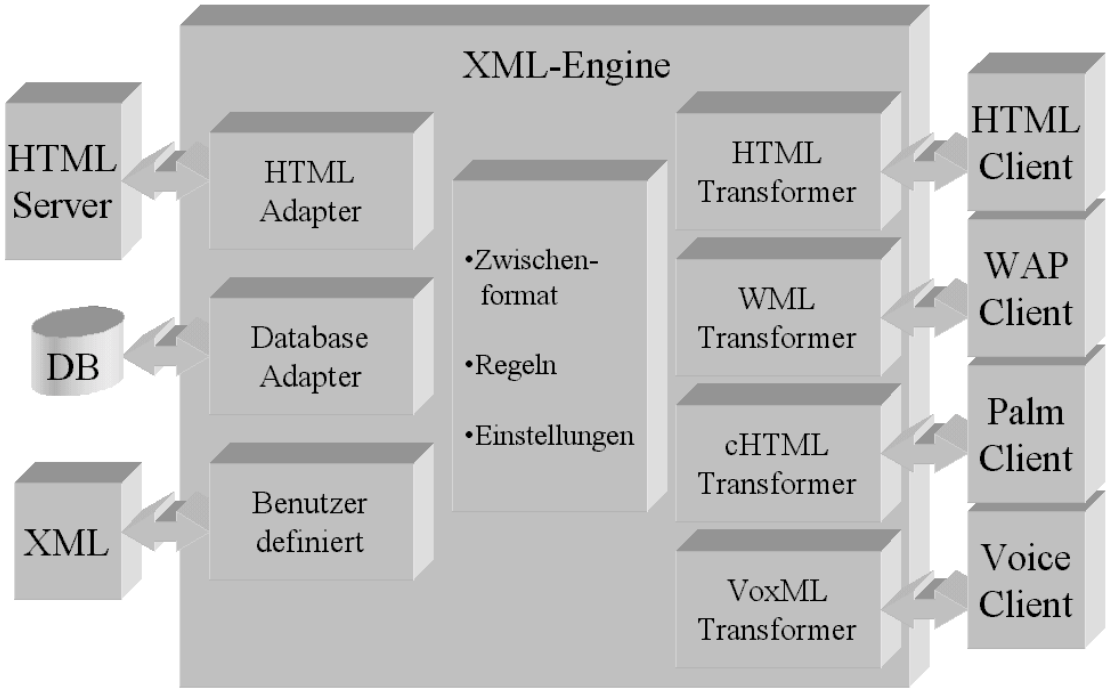


Abbildung 7: Portal-to-Go Architektur

Die Implementierung der Oracle Lösung basiert auf Oracle-eigenen Technologie Oracle8i, XML und Java, was eine Anbindung an andere Technologien vereinfacht.

2.7.2 Prism von Spyglass

Prism unterstützt folgende Arten von Konvertierung:

1. Automatische Text- und Bild-Konvertierung
2. Inhaltsextraktion
3. Konvertierung von Markup-Sprachen
4. Benutzerdefinierte Konvertierung

Die erste Konvertierungsart kann verschiedene Textformate als Eingabe verarbeiten, als Ausgabeformat werden nur HTML und WML unterstützt, außerdem können verschiedene Bildformate manipuliert werden. So kann die Auflösung und die Farbtiefe geändert sowie die

Formatkonvertierung vorgenommen werden. Das WBMP-Format wird unter anderem auch unterstützt.
Die zweite Konvertierungsart dient zum Entfernen von Teilen des Quelldokumentes oder zum Zurückliefern von Inhalten. Diese Option erfordert sowohl die Kenntnisse über das Format als auch über die Struktur des Dokumentes.
Die dritte Möglichkeit erlaubt eine Übersetzung einer Markup-Sprache in eine andere, z.B. die Ersetzung der HTML-Tags in die entsprechende WML-Tags, entfernen von HTML-Tags, die keinen entsprechenden WML-Tag haben, oder die Ergänzung mit schließenden Tags, falls in HTML die schließenden Tags fehlen.
Die letzte Konvertierungsart erlaubt das Entwickeln von eigenen Konvertierungen in C oder C++.
Die Architektur von Prism basiert auf CORBA und erlaubt das Benutzen von Microsoft IIS und Netscape Enterprise Server auf Solaris und NT. Als Ausgabeformate werden z.Z. nur HTML und WML unterstützt.

2.7.3 TRANSWAP

Ein anderer Ansatz wird von TRANSWAP verfolgt:
Es wird für die HTML-WAP Konvertierung zwei neue Elemente, zwei Kommentare, hinzugefügt, die die Teile des HTML-Dokumentes markieren, die nach WML konvertiert werden sollen.
Der Nachteil dieses Verfahrens besteht darin, dass nur einfache Textinhalte dargestellt werden können.

2.8 Architektur des infoAsset Brokers

Bei dem Web-Portal, das um ein WAP-Portal erweitert werden soll, handelt es sich um den infoAsset Broker. Der infoAsset Broker basiert auf einer multi-tier Client-Server-Architektur. Das System ist Web-basiert und setzt auf HTTP zum Datenaustausch auf. Auf Client-Seite werden HTML-Browser eingesetzt, was die Plattformunabhängigkeit für den Endbenutzer sichert. Die zugrundeliegenden Technologien sind XML, HTML und Java, wodurch eine Anpassung an andere Systeme problemlos möglich ist.
Die Mehrschichtarchitektur des Brokers bietet auch eine einfache Erweiterbarkeit und Anpassung an die Anforderungen des jeweiligen Projektes.

Die Schichten, aus denen der Broker besteht, werden in Java-Packages reflektiert (im Bild von unten nach oben):

- Das `util`-Package enthält einige Hilfsklassen, wie z.B. die Klassen `XSLTProzessor`, `MailSender`, etc.
- Das `store`-Package abstrahiert von der Persistenz und stellt vereinheitlichte Speicherungsoperationen für die oberen Schichten zur Verfügung.
- Das `services`-Package stellt verschiedene Dienste zur Verfügung. Diese Dienste implementieren Objekte und Operationen auf Objekte, die im Package `interfaces` spezifiziert werden.
- Das Package `interfaces` spezifiziert die Objekte und die Operationen auf diese Objekte in Form von Schnittstellen. Diese Objekte sind z.B. Assets, die eine Informationseinheit und Grundstrukturen für andere Objekte bilden. So hat z.B. ein

Asset eine Funktion `getId()`, die Id eines Assets zurückliefert. Diese Funktion muss in allen Klassen vorhanden sein, die den Interface `Asset` implementieren.

- `session` stellt Klassen zur Verfügung, die für Session-Tracking, Client-Identifizierung und Verwaltung sowie Substitution verantwortlich sind.
- Die darüber liegende Handler-Schicht bedient sich der Klassen in den unteren Schichten. Im `handler`-Package ist die Anwendungslogik des Systems gekapselt.
- Eine weitere Schicht, die jedoch nicht mehr durch Java-Packages reflektiert wird, bilden die HTML-Templates. Es sind vorgenerierte HTML-Seiten, die mit Hilfe der Handler dynamisch mit dem Inhalt gefüllt werden.
- Die oberste Schicht bildet die `server`-Schicht, die Anfragen vom Client entgegennimmt und die Parameter, sowie HTTP-Header aus den Anfragen extrahiert.

Die Abbildung 8 stellt die Software-Architektur des infoAsset Broker dar:

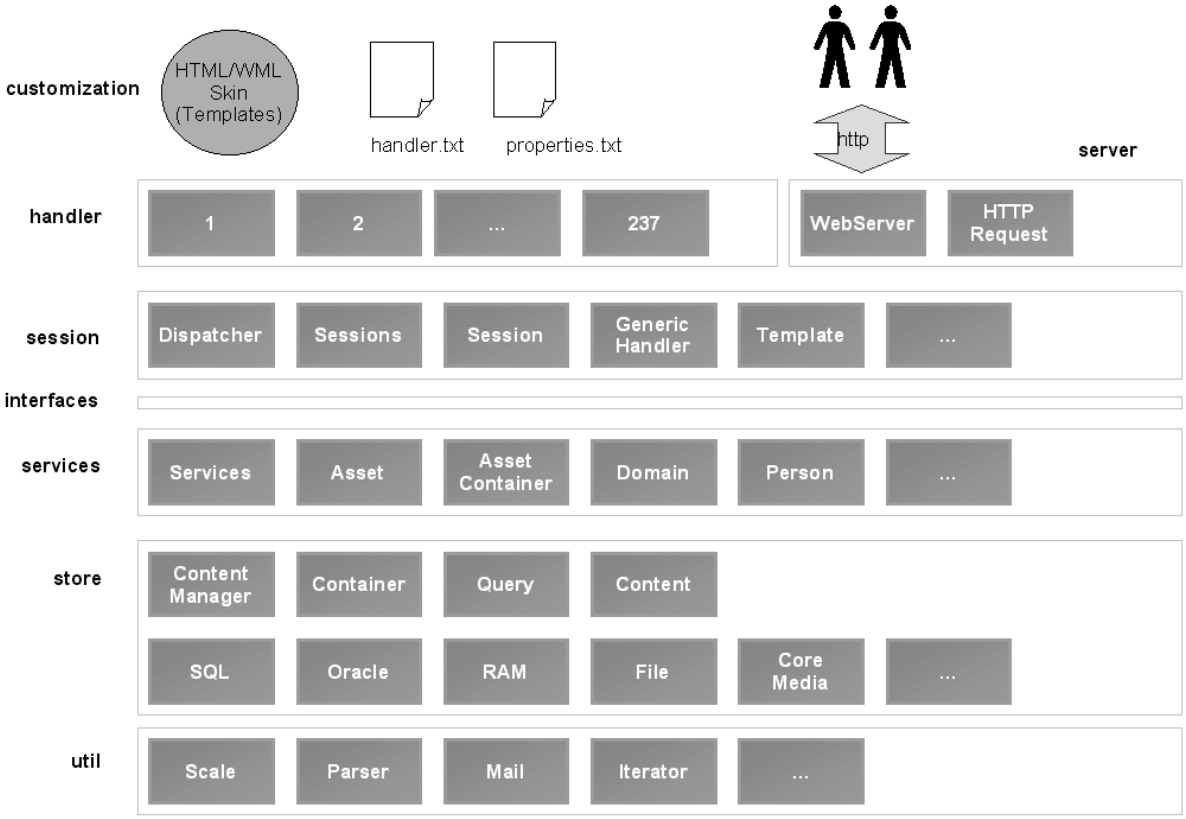


Abbildung 8: infoAsset Broker-Architektur [Quelle: infoAsset AG]

Jedes Package kann mit zusätzlichen Klassen erweitert werden, ohne dass dabei die bestehenden Klassen geändert werden müssen. Falls z.B. eine neue Funktionalität hinzugefügt werden soll, wird eine neue Klasse dem `services`-Package hinzugefügt und neue Handler und Templates geschrieben, um die Funktionalität dem Benutzer verfügbar zu machen.
Die grundlegende Mechanismen wie Session-Tracking, Substitution und Handler, die zum Verständnis der Funktionsweise des Brokers beitragen und für die Realisierung der WAP-Erweiterung notwendig sind, werden im folgenden Kapitel kurz beschrieben.

2.8.1 Session-Tracking

Da der infoAsset Broker Multi-User-Fähigkeit besitzt, ist es nötig, neben den übrigen Vorkehrungen bzgl. multi-threading, einen Mechanismus zu erläutern, der die Zuordnung der Clients zur aktuellen Sitzung implementiert. Dieser Mechanismus wird auch als Session-Tracking bezeichnet. Die Abbildung 9 verdeutlicht die Zuordnung der Clients zu den Sitzungen und den Benutzern:

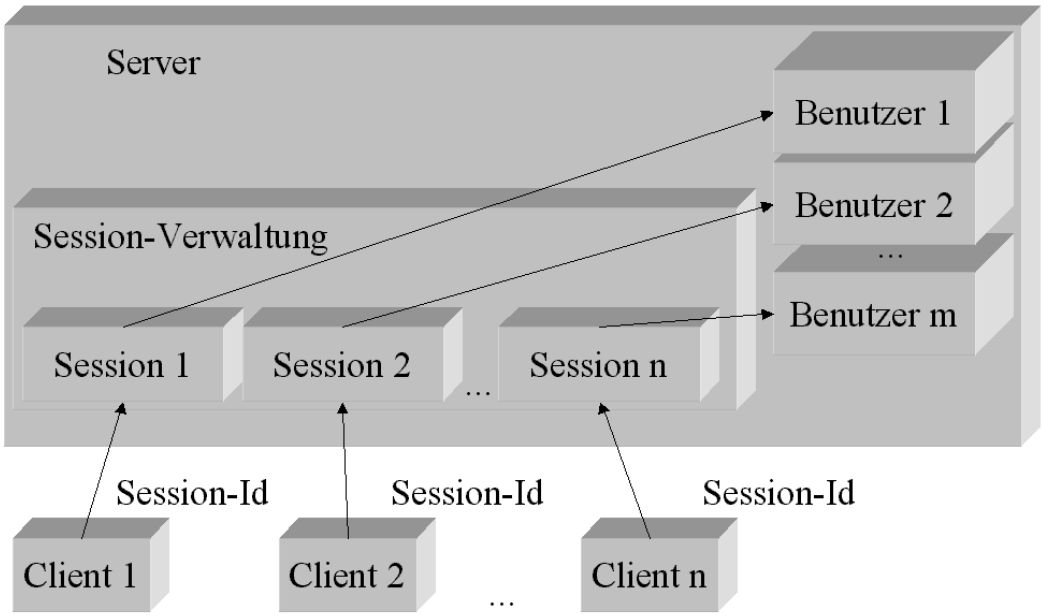


Abbildung 9: Session-Tracking

Für jede neue Sitzung wird vom Server eine neue Session-Id generiert und an den Client übermittelt. Diese Zahl ist zufällig und ausreichend groß, so dass sie eindeutig und schwer zu erraten ist. Die Session-Id wird für die Dauer einer Sitzung bestehen und bei jeder Anfrage vom Client an den Server gesendet, um somit den Client gegenüber dem Server eindeutig zu identifizieren. Die Session-Id wird in Form eines get-Request-Parameters übermittelt und vom Server aus dem URL-String extrahiert.

2.8.2 Substitution

Die Substitution ist ein wichtiger Mechanismus, mit dessen Hilfe eine Generierung von dynamischen Seiten möglich ist. Dabei wird nicht die ganze Seite dynamisch generiert, wie es beispielsweise mittels XSLT der Fall wäre, sondern es werden mit Hilfe eines HTML-Editors vorgefertigte Seiten, Skins oder HTML-Templates, mit Platzhaltern verwendet. Die Platzhalter werden vor der Auslieferung an den Client substituiert. Es gibt fünf verschiedene Substitutionsarten, die auch als Vorlagen bezeichnet werden:

- Einfache Substitution
- Bedingte Substitution (`ConditionalTemplate`)
- Listenvorlage (`ListTemplate`)
- Seitenlistenvorlage (`PageListTemplate`)
- `include`-Platzhalter

Eine Substitution besteht aus zwei Komponenten, eine davon ist ein Platzhalter in einem Template, in das ein Inhalt eingesetzt werden soll. Eine andere Komponente ist eine entsprechende Funktion im Handler, die die Platzhalter findet und diese z.B. im Falle einer einfachen Substitution durch eine Zeichenkette ersetzt.
Bedingte Substitution dient dazu, eine Substitution, in Abhängigkeit von einer Bedingung, die vom Server vorgegeben wird, durchzuführen. Ein Beispiel dafür ist die Anzeige einer Information in Abhängigkeit von der Gruppenzugehörigkeit des Benutzers.
Die Listenvorlage ermöglicht das Generieren von verschiedenen Sorten von Listen, deren Größe erst zur Laufzeit bekannt ist.
Die Seitenlistenvorlage ermöglicht das Aufteilen von langen Listen auf mehrere Listen mit vorgegebener Größe.
Eine Seitenlistenvorlage darf nicht in einer anderen Vorlage enthalten sein. Sie darf jedoch andere Vorlagen (`ConditionalTemplate`, `ListTemplate`) enthalten.

Das folgende Beispiel zeigt eine einfache und eine bedingte Substitution:

Datei mit zwei Platzhaltern, die substituiert werden:

```
Dass ist eine einfache Substitution:
$simple$
Und das ist eine Bedingte Substitution:
$[cond$ If-Zweig $]cond[$ Else-Zweig $cond]$
```

Dabei wird der Platzhalter `$simple$` durch den vom Server vorgegebenen Inhalt ersetzt.
Die Anweisungen der letzten Zeile, enthalten die Platzhalter der bedingten Substitution, bei der je nach Erfüllung einer Bedingung, die serverseitig ausgewertet wird, der ***If-Zweig*** oder der ***Else-Zweig*** eingeblendet werden, wobei innerhalb eines der Zweige andere Arten von Templates enthalten sein dürfen.

2.8.3 Handler

Die Handler bilden zusammen mit den Templates die Anwendungslogik. Beim Aufruf einer Seite können folgende drei Fälle eintreten:

- Es wird eine unsichtbare Seite aufgerufen.
- Es wird eine sichtbare Seite aufgerufen.
- Es wird eine statische Seite aufgerufen.

Bei Aufruf einer Seite, ob sichtbar oder unsichtbar, wird ein Handler aufgerufen. Der Handler wertet die Parameter aus, die mit der Anfrage von dem Client gesendet werden und für die Verarbeitung notwendig sind, lädt das Template und führt die Substitution aus mit den Werten, die sich aus der Verarbeitung ergeben haben. Zum Beispiel kann der Handler bei der Anfrage einer Seite, die eine Liste der Dokumente in einem Unterverzeichnis zeigen soll, folgende Schritte durchlaufen:

1. Das Template, das für die Darstellung der Listen von Dokumenten zuständig ist, wird geladen.
2. Die URL-Parameter, wie Id des Unterverzeichnisses, werden eingelesen.
3. Alle Dokumente des Unterverzeichnisses werden eingelesen.
4. Die Listensubstitution wird durchgeführt, d.h. der Platzhalter wird durch die Liste ersetzt.

5. Alle anderen Substitutionen werden durchgeführt, unter anderem auch die Session-Id und andere Statusinformationen eingefügt.
6. Die fertige Seite wird an den Client gesendet.

Im Falle einer unsichtbaren Seite wird kein Template substituiert, sondern es wird eine Aktion ausgeführt. Diese Aktion kann z.B. das Anlegen eines neuen Benutzers oder Löschen eines Dokumentes sein. Nach einer Aktion wird eine Weiterleitung aktiviert. Dies ist gleichzusetzen mit dem internen Aufruf einer sichtbaren Seite.

Die Abbildung 10 zeigt den Unterschied zwischen dem Aufruf einer sichtbaren und einer unsichtbaren Seite:

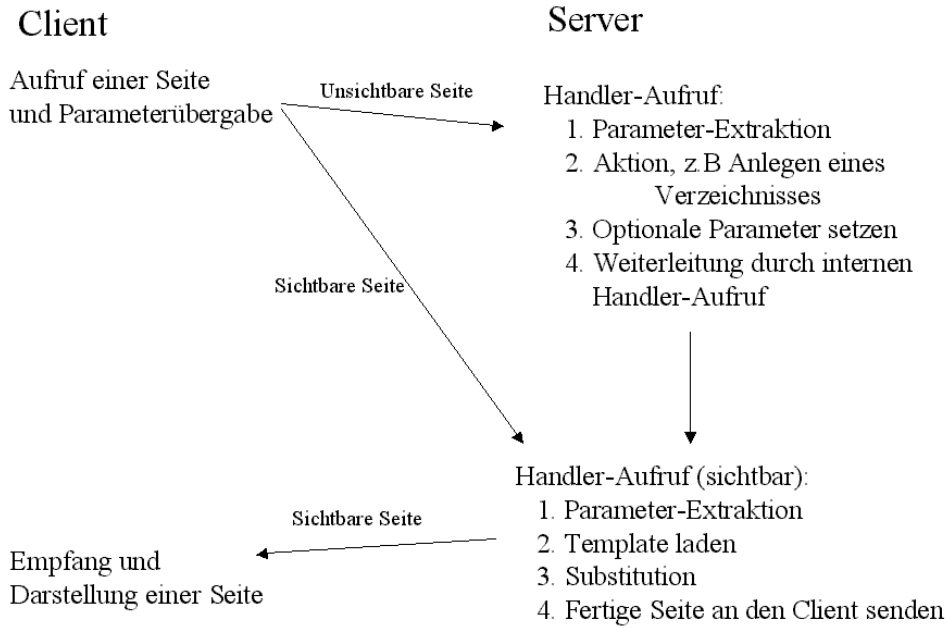


Abbildung 10: Handler

Die Programmlogik entsteht dadurch, das der Benutzer auf der Seite, die er vor sich hat, eine Auswahl von Links hat. Durch das Aktivieren eines der Links wird eine serverseitige Verarbeitung ausgelöst, die durch die Handler vorgegeben ist. Durch die Realisierung der Benutzerschnittstelle durch Handler und Templates entsteht die Möglichkeit, die Gestaltung der Seiten getrennt von der Anwendungslogik durchzuführen. Die Vorteile dieser Vorgehensweise werden im nächsten Kapitel behandelt.

2.8.4 Generelle Trennung der projektspezifischen Oberfläche von generischer Software

Im Kapitel 2.2 wurde die Schichtenarchitektur beschrieben. Durch die Trennung der Anwendungslogik von den darunter liegenden Schichten, die die Grundfunktionen bereitstellen, ist es möglich, die verschiedenen Projekte auf der Basis einer generischen Anwendung zu entwickeln. Dazu gehört auch, dass eine Anwendung, die auf Mehrschichtarchitektur basiert, sehr einfach um eine neue Client-Klasse erweitert werden kann, da im Idealfall nur neue Templates etnwickelt werden müssen, die Handler aber weiterverwendet werden können.

Durch die Trennung der Benutzeroberfläche, die in den Templates realisiert wird, ist es von der Anwendungslogik möglich, dieselben Handler für verschiedene Projekte zu verwenden. So kann man den Funktionsumfang z.B. durch das Entfernen von Links oder Schaltflächen aus dem Template reduzieren, ohne am Programmcode etwas zu ändern und neu zu kompilieren. Auch die Erweiterung ist einfach: durch Hinzufügen von neuen Links und entsprechender Handler und Templates können neue Funktionen realisiert werden, ohne die vorhandene Software zu modifizieren.

2.8.5 Funktionsumfang des infoAsset Brokers

Der infoAsset Broker bietet einen Satz von Funktionen, der erweitert und an die projektspezifischen Anforderungen angepasst werden kann. Die Grundfunktionen haben folgenden Umfang:

- User Profiling ist die Benutzerverwaltung mit der Möglichkeit, die Benutzerdaten zu editieren, die Benutzer verschiedenen Gruppen zuzuordnen, neue Benutzer anzulegen und zu löschen.
- Gruppenverwaltung
- Sammelmappen
- Dokumente
- Dateien
- Verzeichnisse
- Suchfunktion
- Zugriffsberechtigungsfunktion auf Benutzer und Gruppenebenen für Dokumente, Verzeichnisse, Dateien
- Personalisierung, Speicherung von benutzerdefinierten Daten, Dokumenten und Dateien
- Pinnwand, etc.

3 WAP und WML, Grundlagen der Arbeit

3.1 WAP-Forum

Das WAP-Forum wurde 1997 von einer Reihe von Firmen gegründet, die es sich zum Ziel gesetzt haben, einen Standard zu definieren, der nahtlos an die vorhandenen Internetstrukturen anschließt und gleichermaßen die Entstehung vieler inkompatiblen herstellereigenen Formate verhindert. Das Forum hat ein Protokoll geschaffen, das auf den bestehenden Internetstrukturen aufsetzt und den Grundgedanken des Internets auf das mobile Internetstandard mit allen seinen Vor- und Nachteilen überträgt.
Die Vorteile von WAP und von mobilen interaktiven Diensten im allgemeinen sind folgende: der Zugriff auf benötigte Informationen ist überall und stets gegeben, sofortige Reaktion auf Ereignisse, situationsbezogene Dienste, wie Staumeldungen sind stets aktuell zur Hand. Mit der Entwicklung ortsbezogener Dienste kann man auch die Informationen an den Ort binden, an dem sich gerade derjenige befindet, der die benötigte Information abfragt. Unified Messaging erlaubt z.B. das Umleiten der Informationen aus einer E-Mail auf ein Faxgerät.
Die Nachteile des Standards liegen zu einem in der Natur der mobilen Geräte selbst, zum anderen in technologischen Schranken (s. Kapitel 2.4). Die GSM-Netze haben sehr geringe Bandbreite von 9,6 KBit/s. Außerdem verursacht der lange Verbindungsaufbau nach längeren Benutzungspausen (Time-out) sehr hohe Latenzzeiten. Mit neuen paketvermittelnden Verfahren ist man immer online, so dass die Einwahlprozedur entfällt. Dabei erhöhen sich auch die Durchsatzraten und bandbreitenintensive Anwendungen werden ermöglicht.
Ein weiterer Faktor, der sich bei den mobilen Telefonen negativ auswirkt, ist die begrenzte Möglichkeit der Benutzerschnittstelle. Die Tastatur besteht nur aus Ziffern, was zwar durch die intelligente Eingabeprogramme wie T9, kompensiert wird, jedoch bei Login- und Passworteingaben oder anderen Anmeldeprozeduren ist man auf das mehrfache Betätigen einer und derselben Taste angewiesen. T9 vermeidet das Mehrfachbetätigen der Nummerntasten, die dreifach bis vierfach mit Buchstaben beleget sind, durch den Vergleich der schon betätigten Tasten mit allen möglichen dem Zahlenmuster entsprechenden Wörtern im Wörterbuch.
Ebenso ist die Displayanzeige auf maximal 5 Zeilen begrenzt, was zur Folge hat, dass die geringe Größe des Displays keine vernünftige Formatierung der Texte zulässt. Eine weitere Beschränkung ist die Farbtiefe der Bilder, die meistens auf ein Bit limitiert ist, so dass nur schwarz-weiß Bilder angezeigt werden können.

3.2 WAP-Spezifikation

3.2.1 WAP-Schichtenmodell

WAP-Standard besteht aus einer Reihe von Spezifikationen und basiert auf einem Schichtenmodell. Die fünf Schichten [Imm2000] sind folgende:

Schichtname	WAP-Protkoll
Anwendungsschicht (Application Layer):	WAE (Wireless Application Environment)
Sitzungsschicht (Session Layer)	WSP (Wireless Session Protocol)
Übertragungsschicht (Transaction Layer)	WTP (Wireless Transaction Protocol)

Sicherheitsschicht (Security Layer)	WTLS (Wireless Transport Layer Security)
Transportschicht (Transport Layer)	WDP (Wireless Datagram Protocol)

Tabelle 1: WAP-Schichtenmodell

Folgendes Bild zeigt das WAP-Schichtenmodell [SCNEd2000] im Vergleich zu Internet- und OSI-Schichtenmodell:

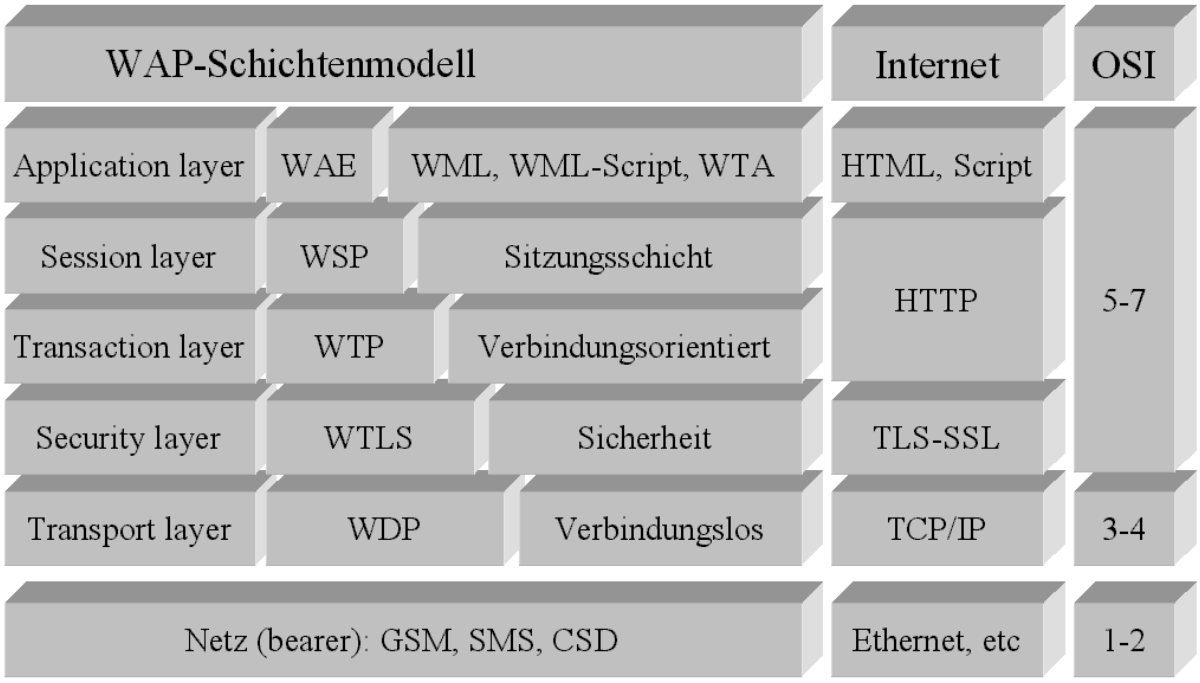


Abbildung 11: WAP vs. Internet vs. OSI

Die Anwendungsschicht (Application Layer) ist in einem Micro-Browser des User-Agents implementiert und beinhaltet WML (Wireless Markup Language), WMLScript (eine JavaScript ähnliche Skriptsprache) und WTA (Wireless Telephony Application). Diese drei Komponenten können benutzt werden, um eine interaktive WAP-basierte Anwendung zu entwickeln. Dabei werden die WMLScript und WTA-Funktionen aus WML heraus aufgerufen. Die Datenvorverarbeitung kann mit Hilfe von WMLScript auf dem Client realisiert werden (Daten-Validierung, clientseitige Berechnung ohne Serveranfrage, etc). Mit WTA hat man einen Mechanismus geschaffen, mit dessen Hilfe Anrufe von einer WML-Seite betätigt werden können (sehr praktisch für ein WAP-basiertes Telefonauskunftsystem). Auch die Übernahme von Telefonnummern in das Telefonbuch des mobilen Endgerätes ist möglich.

Das Wireless Session Protocol (WSP) bietet eine Schnittstelle für die Anwendungsschicht, die es erlaubt, zwei Arten von Sitzungsdiensten zu etablieren. Die erste ist ein verbindungsorientierter Dienst oberhalb von WTP, die zweite ist ein verbindungsloser Dienst oberhalb des ungesicherten oder gesicherten Datagramm-Dienstes (WDP).

Das Wireless Transaction Protocol (WTP) befindet sich über einem Datagram-Service (hier WDP) und ist ein light-weight verbindungs-orientiertes Protokoll, das für die Ausführung von als zuverlässig und unzuverlässig deklarierten Transaktionen sorgt.

Die Wireless Transport Layer Security (WTLS) ist die Sicherheitsschicht des WAP, die technisch gesehen auf dem SSL-Nachfolger TLS [Imm2000] basiert. WTLS dient zur Sicherung von:

- Datenintegrität: WTLS stellt sicher, dass die Daten unverändert zwischen zwei Kommunikationspartnern ausgetauscht werden.
- Privatsphäre: WTLS stellt sicher, dass die Daten, die ausgetauscht werden, nicht von anderen, sondern nur von dem Kommunikationspartner interpretiert werden können.
- Authentifizierung: Sicherheitsschicht ermöglicht das Anmelden und das Identifizieren des Kommunikationspartners.
- Schutz vor Denail-of-Service-Attacken durch Erkennung von wiederholten oder nicht verifizierten Daten.

Das Wireless Datagram Protocol (WDP) ist die Transportschicht und stellt eine Schnittstelle zwischen dem Bearer und den darüber liegenden Schichten. Als Bearer wird die Schnittstelle zwischen WAP und physikalischen Netzen bezeichnet. WDP stellt also sicher, dass alle vorhandenen Netze auf gleiche Weise angesprochen werden und gleiches Verhalten nach oben hin zeigen, ungeachtet, ob es sich um SMS, GSM oder andere Netzdienste handelt.

Die verschiedenen Schichten des WAP müssen im WAP-Gateway und in Endgeräten implementiert werden, um die verschieden Arten der Verbindung zu erhalten [Areh2000]:

Art der Verbindung	Verwendete Protokolle
Nicht gesichert verbindungsorientiert	WSP-WTP-WDP
Gesichert verbindungsorientiert	WSP-WTP-WTLS-WDP
Gesichert verbindungslos	WSP-WDP
Nicht gesichert verbindungslos	WSP-WTLS-WDP

Tabelle 2: Verschieden Arten von Verbindungen in WAP

Durch das Schichtenmodell des WAP ist es möglich, eigene Anwendungen zu entwickeln, die außerhalb von WAP-Spezifikationen liegen und dort auf einer der Schichten aufsetzen, die am besten für die Anwendung geeignet ist.

Die Transportschicht beinhaltet die Schnittstelle zwischen WDP und dem Mobilfunknetz. Als Trägernetz kann jedes der heute vorhandenen digitalen Mobilfunknetze eingesetzt werden. Somit ist WAP auch für zukünftige Netztechnologien gerüstet, da nur die Transportschicht an die neuen Netze angepasst werden muss. WAP unterstützt folgende Trägernetztechnologien wie GSM, GPRS, HSCSD, SMS, CSD, CDPD, etc.

Der User Agent beinhaltet in der Anwendungsschicht beschriebene Komponenten wie WTA, WML und WMLScript und andere geräteherstellerspezifische Funktionen. User Agent implementiert eine Zustandsmaschine, History, Cache und verwaltet Variablen auf WML-Ebene. Die Verwendung von History und Cache wurde von HTML übernommen. Darüber hinaus bieten die Variablen einen zusätzlichen Mechanismus, um den Client-Status zu verwalten. Mit WMLScript und WTA wird das Spektrum der möglichen Anwendungen stark erweitert. Diese bieten die Möglichkeit, die Daten clientseitig zu verarbeiten und die Telefonie-Funktionen aus WML oder WMLScript zu aktivieren.

3.2.2 WAP-Distributionsmodell

Um WAP zu verstehen, ist es wichtig, sich zu veranschaulichen, wie die Daten vom Server zum mobilen Gerät kommen. Die WAP-Anwendungen werden von normalen Internetservern, auch Origin-Server genannt, zur Verfügung gestellt. Dabei kann eine Web- und eine WAP-Anwendung auf demselben Server installiert sein [Imm2000].

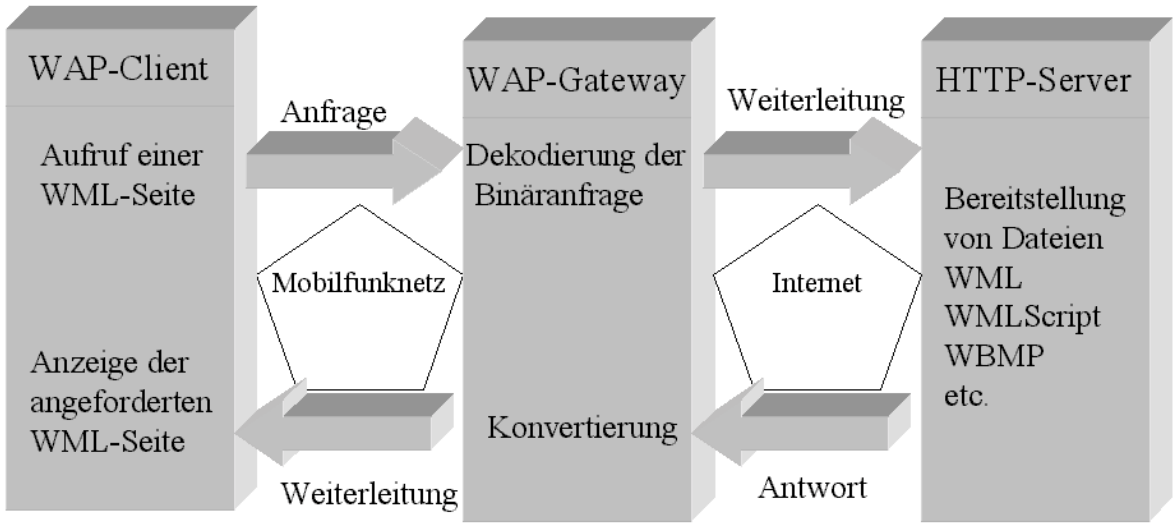


Abbildung 12: WAP-Distributionsmodell

Die jeweiligen MIME-Types müssen eingetragen sein. Die MIME-Types und die entsprechenden Dateitypen sind folgende [Imm2000 und Areh2000]:

MIME-Types	Datei-Typen
text/vnd.wap.wml .wml	WML
test/vnd.wap.wmlscript .wmls	WMLScript
text/vnd.wap.wmlscript .wmlscript	WMLScript
application/vnd.wap.wmlc .wmlc	WML kompiliert
application/vnd.wap.wmlscriptc .wmlsc	WMLScript kompiliert
image/vnd.wap.wbmp .wbmp	Bilder im WBMP-Format
text/x-vcard .vcf	Visitenkarte im VCF-Format
text/x-vCalendar .vcs	Kalender/Termin-Datei im VCS-Format
text/x-vmel .mel	Klingelton im Mel-Format von Ericsson

Tabelle 3: WAP-MIME-Types

In der Mitte des Übertragungsweges befindet sich das WAP-Gateway, ein Proxy-Server, der das mobile Netz mit dem Web verbindet. Das WAP-Gateway wird meistens von den Netzanbietern zur Verfügung gestellt und leistet eine Reihe von Diensten, in erster Linie die Umsetzung der in Textform vorliegenden WML und WMLScript Dateien in das platzsparende binäre Format WBXML (WAP Binary XML).
Dabei wird die Information, mit Ausnahmen von redundanden Informationen wie Kommentaren, verlustfrei komprimiert.
Hier ist eine Übersicht der Dienste, die vom WAP-Gateway geleistet werden können [Imm2000, Areh2000 und Risch2000]:

- Die Konvertierung der Bilder in das WAP-spezifische WBMP-Format. WBMP-Format stellt die Möglichkeit dar, schwarz-weiß Bilder auf einem mobilen Gerät anzuzeigen.
- Übersetzung von HTML-Seiten in WML
- Puffern der Dateien
- Zugriffskontrolle
- Sicherheitskontrolle
- Zeichensatzkonvertierung, falls der User Agent den vom Origin-Server gelieferten Zeichensatz nicht unterstützt.
- Konvertierung des WSP in HTTP. Die Konvertierung ist notwendig, um der geringen Bandbreite und der hohen Latenz der Mobilfunknetze gerecht zu werden.
- Kompilierung des WMLScript
- Konvertierung der WML-Dateien in Binärformat

Hier ist ein Beispiel für die Übersetzung eines Fragmentes einer WML-Datei in das WBXML-Format [WML2000]:

```
<wml>
<card id=“abc" ordered=“true“>
<p>
<do type="accept">
...
```

Dieser WML-Code-Auszug wird in folgende binärkodierte Form (Hexadezimalzahlen) konvertiert:

```
7F E7 55 03 'a' 'b' 'c' 00 33 01 60 E8 38 01
```

Die Tabelle enthält die Beschreibung der einzelnen im Beispiel verwendeten Token:

Token	Bedeutung
7F	wml-Element mit Kontent
E7	card-Element mit Kontent
55	id=
03	Textanfang
'a' 'b' 'c' 00	abc + Terminierungszeichen
33	ordered="true"
01	Ende der Attribut-Liste des card-Elementes
60	p-Element
E8	do-Element mit Kontent
38	type=accept
01	Ende der Attribute Liste des do-Elementes

Tabelle 4: WBXML-Token und ihre Bedeutung

Auf der Mobilfunkseite kommuniziert das WAP-Gateway mit dem mobilen Gerät mittels des WAP-Stapels, der die zu den im Internet eingesetzten komplementären Protokolle nutzt.

Die WAP-Anwendung wird auf einem Server, der auch gleichzeitig eine Web-Anwendung bereitstellen kann, installiert. Einige Konzepte des WWW wurden in das WAP-Modell übernommen. Dazu gehört auch die Verwendung der URLs, um die Inhalte auf dem Origin-Server zu referenzieren. Es werden auch MIME-Types zur Unterscheidung der verschiedenen Datei-Typen eingesetzt. Außerdem besitzt WAP analog zum Web die Markup-Sprache WML,

eine clientseitige Skriptsprache, WMLScript, und unterstützt Bilder sowie andere Formate wie VCF und VCS, auf die in Kapiteln 3.2.7.1 und 3.2.7.2 eingegangen wird.

Das dritte Glied in der Kette ist das mobile Endgerät oder auch User-Agent genannt. Dies kann sowohl ein mobiles Telefon, ein PDA als auch ein Software-WAP-Emulator sein. Die auf dem Markt befindlichen Geräte unterstützen die WAP-Version 1.1, die WML, WMLScript und WTA beinhaltet. Außerdem können die Geräte verschiedene herstellerspezifische Dateiformate verarbeiten, wie z.B. Ericssons Klingelton-Dateien. Nähere Beschreibung der Endgeräte und deren Funktionen erfolgt in den nächsten Kapiteln.

3.2.3 WML

WML wurde entwickelt, um den Beschränkungen der mobilen Geräte und der Funknetze gerecht zu werden. Diese Beschränkungen beinhalten:

- Kleine Displays und begrenzte Ausgabemöglichkeiten
- Geringe Bandbreite der Netze
- Begrenzte Speicher und Rechenressourcen

Die Aufgaben von WML sind mit denen von HTML identisch und schließen folgendes mit ein:

- Textdarstellung und Layout: WML unterstützt Bilder, Textformatierungs-Tags und Attribute, die z.B. Ausrichtung, Schriftart und Verhalten von Text wie Zeilenumbruch beinflussen.
- Aufteilung der Informationseinheiten in Decks und Karten: Dabei enthält eine WML-Datei jeweils ein Deck, das in Karten, in kleinere logische Einheiten, die jeweils einer Benutzerinteraktion entsprechen, aufgeteilt werden. Durch eine Reihe von Karten, die zu einem Deck gehören, kann lokal auf dem User Agent ohne Serveranfragen navigiert werden.
- WML stellt einen Mechanismus zur Navigation zwischen Karten und Decks bereit und bedient sich der Links in Analogie zu HTML.
- WML-Variablen können innerhalb eines Decks deklariert werden. Der Variablenname wird durch den Wert der WML-Variablen vor dem Anzeigen des Decks ersetzt.
- Kommentare in WML sind XML-konform, z.B. <!-- ein Kommentar -->. Sie werden bei der Binärcodierung ignoriert.
- XML-konforme Unterstützung von verschiedenen Zeichensatzcodierungen

WML unterstützt UTF-8, UTF-16, UCS-4. Auch alle anderen Zeichensätze, die eine Untermenge von UCS-4 sind, wie US-ASCII, ISO-8859-1, können verwendet, müssen aber im Dokumentenprolog deklariert werden [Imm2000]. Dagegen müssen UTF-8 und UTF-16 nicht explizit ausgewiesen werden. Falls der UA einen Zeichensatz nicht unterstützt, kann eine Konvertierung durch das WAP-Gateway vorgenommen werden. Alternativ dazu kann der Origin-Server darauf angemessen reagieren, in dem er den HTTP-Header Accept-charset auswertet und WML-Seiten mit passender Codierung ausliefert.
Zum Benutzen der Sonderzeichen kann die WML-Schreibweise, ähnlich zu HTML, oder Unicode eingesetzt werden.

Sonderzeichen werden folgendermaßen codiert:

Zeichen	WML-Code	Unicode	Beschreibung
<	<	&#060	Kleiner als-Zeichen
>	>	&#062	Größer als-Zeichen
&	&	&#038	Kaufmännisches Und-Zeichen
$	$$		Dollarzeichen

Tabelle 5: Sonderzeichen und ihre Codierung

Das Dollarzeichen nimmt eine Sonderstellung ein, da es für die Variablendeklaration `$var` verwendet wird und muss als `$$` geschrieben werden, falls ein Dollarzeichen angezeigt werden soll.
Die sprachspezifischen Sonderzeichen können direkt als solche im Quellcode erscheinen, falls der UA sie unterstützt.
WML ist eine XML-Anwendung. Das bedeutet, dass in XML geltende Regeln auch in WML-Dokumenten eingehalten werden müssen:

- Groß- und Kleinschreibung ist relevant.
- Jeder Tag muss geschlossen werden (`<tag> inhalt </tag>` oder `<tag/>`).
- Attribute müssen von `"` oder `'` umschlossen sein (`<tag attr="abc"/>`).
- Jedes Dokument fängt mit einem Prolog an.

Hier ist ein kurzes Beispiel, das die Grundstruktur eines WML-Dokumentes mit Prolog, zwei Karten und Links zeigt:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="card1" title="Erste Card">
<p>
Das ist die Erste Card
<br/><a href="#card2">zur Zweiten Card</a>
</p>
</card>
<card id="card2" title="zweite Card">
<p>
Das ist die zweite Card
</p>
</card>
</wml>
```

Es sieht auf einem mobilen Gerät in etwa so aus (hier das Nokia WAP Toolkit 2.0 mit dem Blue Print Modell):

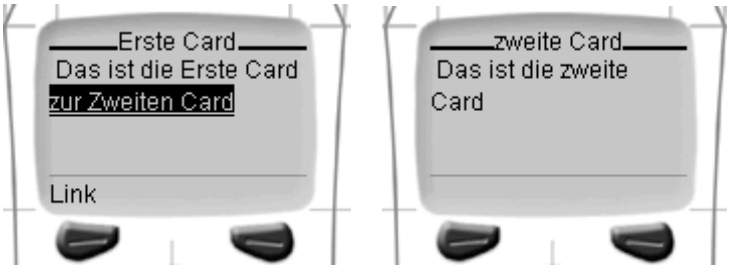


Abbildung 13: WML-Darstellung

Dieses WML-Deck, also eine WML-Datei, enthält zwei Karten. Der eigentliche Inhalt des Decks ist von einem öffnenden und einem schließenden `wml`-Tag umgrenzt. Innerhalb dieser Tags können sich eine oder mehrere Karten befinden. Der `card`-Tag kann ein `id`-Attribut haben. Das `id`-Attribut ermöglicht das Verwenden einer Karte als Ziel eines Links. So kann im obigen Beispiel die zweite Karte mit `id="card2"` mit Hilfe eines Links in der ersten Karte angewählt werden. Dabei ist das Ziel als `href="#card2"` Attribut ähnlich zu den HTML-Labels spezifiziert. Die Karten verhalten sich jedoch etwas anders als die HTML-Labels. Eine HTML-Seite wird als eine Einheit angezeigt und die Labels dienen als Sprungmarken, um langes Scrollen zu vermeiden. Die Karten in WML stellen dagegen eine eigenständige grafische Einheit dar, d.h. es wird nur eine Karte auf einmal angezeigt.
Es gibt eine Limitierung bei der Auswahl der Attributwerte für das `id`-Attribut. Sie müssen innerhalb eines Decks eindeutig sein und mit einem Buchstaben anfangen.
Das zweite Attribut ist das `title`-Attribut, das dazu dient, den Titel der Karte zuzuweisen. Der Titel erfüllt keine logische Funktion und kann beliebige Zeichen enthalten (Dollarzeichen muss als `$$` angegeben werden). Der Titel wird je nach Browser entweder in einer Titelzeile angezeigt oder ignoriert. Das `p`-Element ist obligatorisch, es umschließt alle Inhalte einer Karte. Die Inhalte können auch auf mehrere `p`-Elemente verteilt werden. Die `do`- und `onevent`-Elemente gehören nicht zu den Inhalten und müssen deshalb außerhalb der `p`-Elemente definiert werden. Die Benutzung und die Attribute des `p`-Tags werden im Kapitel 3.2.4.8 beschrieben.

3.2.4 WML-Elemente

3.2.4.1 Variablen

WML erlaubt die Verwendung von Variablen. Der Lebenszyklus der Variablen beschränkt sich auf den aktuellen Browser-Kontext. Unter Kontext versteht man den User-Agent-Zustand, inklusive Variablen, Navigationsverlauf und andere browser-spezifische Informationen, die den Bezug zum Browserzustand haben. Wird eine Seite mittels Bookmarks oder manueller URL-Eingabe angesprungen, wechselt der Browser-Kontext. Bei einem Kontextwechsel muss der Browser den alten Kontext terminieren. Ein Kontextwechsel findet nicht statt, wenn z.B. eine Karte auf einem fremden Server aus dem aktuellen Kontext heraus, d. h. mittels eines Links, angesprungen wird.
Die Variablennamen genügen folgender Syntax:

- `$varname`
- `$(varname)`
- `$(varname:modus)`

Der Variablenname wird an der Stelle, wo sie vorkommen, durch den Wert der Variablen ersetzt.

Die Variablen werden durch Benutzer über WML-Formulare (Kapitel 3.2.4.7) oder mit Hilfe des `setvar`-Tags zugewiesen:

```
<setvar name="varname" value="string"/>
```

Die Variablen können innerhalb von `go`, `prev` oder `refresh` Tags definiert werden. Das Setzen erfolgt jedoch, da es sich bei diesen Tags um die Tasks handelt (Kapitel 3.2.4.3), erst beim Auftreten des entsprechenden Ereignisses.

So beispielsweise wird die Variable, die innerhalb der `go`-Tags spezifiziert wurde, erst dann mit dem neuen Wert belegt, wenn der Link des `go`-Tags aktiviert wurde.
Hier sind einige Beispiele, die das Setzen der Variablen mit verschiedenen Methoden zeigen.

a) Beim Aktivieren eines Links:

```
<go href="link">
<setvar name="varname" value="string"/>
</go>
```

b) Bei Rückwärtsnavigation:

```
<prev>
<setvar name="varname" value="string"/>
</prev>
```

d) Beim Neuinitialisieren aller interner Zustände inklusive der Benutzerschnittstelle:

```
<refresh>
<setvar name="varname" value="string"/>
</refresh>
```

Andere Möglichkeit, einer Variablen einen Wert zuzuweisen, ist die Verwendung von verschiedenen Eingabeelementen wie Texteingabefelder oder Auswahllisten. Diese Methode wird im Kapitel 3.2.4.7 gezeigt.
Um die Variable abzufragen, wird vor dem Variablennamen ein Dollarzeichen vorangestellt:

```
$varname
```

Folgt dem Variablennamen kein Trennzeichen wie Space oder Komma, muss der Variablenbezeichner geklammert werden:

```
$(varname)
```

Hier ist ein Beispiel der Parameterübergabe an den Server mit Hilfe von Variablen:

```
<a href="datei.wml?parameter1=$(varname)">senden</a>
```

Der Wert der Variablen wird an der Stelle, wo der Variablenname erscheint, einfach ersetzt. Die Variablen können an folgenden Stellen eingesetzt werden:

- Text
- Werte der `value`-Attribute
- URL-Strings

Es können keine Elemente und Attribute durch eine Variable ersetzt werden.
Für die Variablen-Substitution gibt es eine Reihe von Ersetzungsregeln, die unter dem Begriff Escaping zusammengefasst sind. Escaping gibt an, ob und wie der Wert der Variablen interpretiert werden soll. Der eigentliche Wert der Variablen ist von dem Escaping-Modus unabhängig.

Folgende Tabelle beschreibt die zur Verfügung stehenden Modi:

Modus	Beschreibung
`$var` oder `$(var)`	URL-Encoding ist kontext-sensitiv, d.h es werden nicht alphanumerische Zeichen nur bei URLs konvertiert. Das ist die Standardeinstellung.
`$(var:e)` oder `$(var:escape)`	Die Zeichenkette wird URL-codiert.
`$(var:unesc)`	Die Zeichenkette wird URL-decodiert.
`$(var:n)` oder `$(var:noesc)`	Variable wird unverändert interpretiert.

Tabelle 6: Escaping-Modi

Die beste Wahl bei der Einstellung des Escaping-Modus ist die Standardeinstellung, aber es gibt einige Fälle, bei denen es sinnvoll ist, ein bestimmtes Verhalten zu erzwingen.

3.2.4.2 Links

Es gibt eine Vielzahl von Möglichkeit einen Link zu definieren. Die einfachste ist es, den `a`-Tag zu benutzen:

```
<a href="link.wml">Ein Link</a>
```

Diese Methode ist die Kurzform der Kombination des `anchor` und `go`-Tags

```
<anchor><go href="link"/>Ein Link</anchor>
```

Der Vorteil des `anchor`-Tags liegt darin, dass anstelle vom `go`-Tag auch `prev`-Tag eingesetzt werden kann und so ein Link zur zuletzt angesprungener Seite ermöglicht wird. Dieses ist hilfreich, falls eine Optionen-Seite mit gleichen Menüpunkten für mehrere Seiten realisiert werden soll.
Ein anderer Vorteil der oben beschriebenen Möglichkeit ist, die Variablen innerhalb des `go`-Tags setzen zu können. Dabei kann z.B. eine den Link aufrufende Seite der nächsten Seite bestimmte Parameter direkt übergeben, ohne den Umweg über den Server zu machen.
Die Links werden von dem User Agent durch Unterstreichen oder durch Klammern besonders hervorgehoben. Es gibt eine Möglichkeit, die Links zu definieren und diese mit Hilfe von Softkeys, den Benutzerschnittstellenelementen, die mit dem HTML-Button vergleichbar sind, aufzurufen. Hier ist ein Beispiel der Definition eines Links durch einen Softkey:

```
<do type="accept" label="Link">
<go href="link.wml"/>
</do>
```

Die Werte der `href`-Attribute können absolute, relative und lokale URLs sein, wie die folgenden Beispiele es verdeutlichen:

- `http://wap.wapportal.de/index.wml`
- `index.wml`
- `../set2/set.wml`
- `#card2`

Zur Parameterübergabe an den Server kann sowohl die get- als auch die post-Methode verwendet werden. Die get-Methode ist der Default-Wert und kann sowohl mit dem `a`-Tag als auch mit dem `go`-Tag verwendet werden:

```
<a href="link.wml?paramer1=wert1&amp;parameter2=wert2">Ein Link</a>
```

Dabei muss bei dem Parametertrennzeichen beachtet werden , dass "`&`" und nicht "`&`" eingesetzt wird, weil ein &-Zeichen laut WML-Konvention nicht als "`&`", sondern als "`&`" codiert werden muss.
Die get-Methode hat den Nachteil, dass nur eine begrenzte Anzahl von Zeichen übertragen werden kann, weil verschiedene User Agents unterschiedliche maximale Längen des URL-Strings unterstützen.
Die post-Methode unterliegt diesbezüglich keinen Beschränkungen, kann aber nur mit dem `go`-Tag verwendet werden:

```
<go href="link.wml" method="post">
<postfield name="var1" value="wert1">
<postfield name="var2" value="wert2">
<postfield name="var3" value="wert3">
</go>
```

Die post-Methode wird von einigen Geräten wie Siemens nicht unterstützt.

3.2.4.3 Tasks

Die Tasks werden durch die Tags `go`, `prev`, `refresh` und `noop` initialisiert. Sie führen einen vom Benutzer oder User Agent initiierte Aktion aus.
Der `noop`-Task führt keine Aktion aus und wird dazu verwendet, die im Template definierten Tasks zu überschreiben. Ein Beispiel verdeutlicht die Verwendung dieses Elementes im Kapitel 3.2.4.6.
Der `prev`-Task ruft die vorherige Seite auf und wird für die Rückwärtsnavigation eingesetzt.
Der `refresh`-Task wird dazu verwendet, die Variablen zu setzen oder die Karte zu aktualisieren.
Der `go`-Task ist das meistverwendete Element. Es wird zum Aktivieren eines Links eingesetzt und kann je nachdem, beim Auslösen welches Ereignisses der Link aktiviert werden soll, innerhalb eines `anchor`-, eines `do`- oder eines `onevent`-Tags definiert werden.
Die Benutzung des `anchor`-Tags wurde im Kapitel 3.2.4.2 behandelt.

3.2.4.4 Softkeys

`do`-Elemente binden die Tasks an die Eingabeelemente des Endgerätes, die Softkeys genannt werden. Ein Softkey kann auf verschiedene Art und Weise von dem Gerätehersteller realisiert werden. Es kann eine Taste unterhalb eines LC-Displays sein, dessen Anzeige und Funktion freiprogrammierbar ist, oder es kann auch eine Taste mit einer Aufschrift sein, die nicht geändert werden kann. Die Funktion solcher Taste kann dabei deaktivierbar oder freiprogrammierbar sein. Ein Softkey kann auch als Grafikelement auf einem druckempfindlichen LC-Display erscheinen. Da die WML-Spezifikation keine Vorgaben für die Softkeys macht, ist es denkbar, die Softkeys z.B. als Stimmenkommandos zu realisieren.
Das `type`-Attribut des `do`-Tages gibt vor, auf welche physikalischen Elemente des Endgerätes der Softkey abgebildet wird. Dabei ist die Abbildung von Gerät zu Gerät sehr unterschiedlich.
Die Benutzung der Softkeys ist folgende:

```
<do type="typeAttribut" label="Label">
<task/>
```

```
</do>
```

Das `type`-Attribute dient lediglich als Orientierung für den Anwendungsentwickler und kann vom User Agent ausgewertet oder ignoriert werden. Das `type`-Attribute kann folgende Werte annehmen:

- Der Wert "`accept`" des Attributes bindet den Task an ein Benutzerschnittstellenelement, mit dem der Benutzer die Bildschirmdaten akzeptieren kann. Ein so definierter Softkey ist mit einem OK-Button zu vergleichen.
- Der `prev`-Wert soll zum Navigieren zur vorhergehenden Karte benutzt werden und ist mit dem Back-Button eines HTML-Browsers zu vergleichen. Das `prev`-Attribut des `do`-Elements darf nicht mit dem `prev`-Tag verwechselt werden. Das `prev`-Attribut bindet einen Task an ein Benutzerschnittstellenelement, wogegen der `prev`-Tag ein Task ist und eine Funktion, nämlich das Navigieren zur vorhergehenden Karte, ausführt.
- Der `option`-Wert fügt einen Task in eine Liste von Elementen ein. Durch die mehrfache Definition des `do`-Tags mit dem `option`-Attribut kann also eine Liste von Tasks aufgebaut werden, die über einen Softkey erreicht werden kann.
- Der `help`-Wert gibt eine Möglichkeit, eine kontextabhängige Hilfeinformation abzurufen.
- Der `delete`-Wert soll dem Benutzer ermöglichen, seine Angabe zu löschen.
- Der `reset`-Wert dient zum Löschen eines Formulars oder zum Zurücksetzen des internen Zustandes des User Agents.
- Der `unkown`-Wert ist herstellerabhängig und entspricht einem `do`-Element ohne `type`-Attribut.
- Es gibt weitere herstellerspezifische Werte des `type`-Attributes, die die Form `vnd.*` hat, wobei `*` den Hersteller und den Typ des Benutzerschnittstellenelementes beschreibt.

Da die meisten Endgeräte maximal zwei bis drei Softkeys haben, werden die verschiedenen do-Types auf diese Benutzerschnittstellenelemente mehrfach abgebildet, so dass es durchaus vorkommen kann, dass z.B. `help`- und `accept`-Type sich identisch verhalten.

3.2.4.5 Events

Die Tasks können auch an Ereignisse, so genannte Events, gebunden werden. Es gibt vier Ereignisse, die ausgelöst werden, wenn der Browser eine bestimmte Funktion ausführt:

1. Das `onenterforward`-Ereignis wird ausgelöst, wenn ein Benutzer einen Link oder einen Softkey aktiviert, der auf diese Karte bzw. auf das Deck zeigt.
2. Das `onenterbackward`-Ereignis wird dann ausgelöst, wenn die Karte bzw. das Deck mittels `prev`-Tasks angesprungen wird.
3. Das `onpick`-Ereignis wird ausgelöst, wenn ein Benutzer ein Element auswählt bzw. dessen Auswahl aufhebt.
4. Das `ontimer`-Ereignis wird nach einer vordefinierter Zeitspanne ausgelöst.

Innerhalb eines Ereignisses können genauso wie innerhalb des `do`-Elements die oben beschriebenen Tasks eingesetzt werden. Diese Tasks werden ausgeführt, sobald das betreffende Ereignis ausgelöst wird.

Generelle Vorgehensweise ist folgende:

```
<card id="card1" title="Erste Card" >
<onevent type="Type">
<task/>
</onevent>
...
</card>
```

Dabei kann `Type` durch die Werte `ontimer`, `onenterforward`, `onenterbackward` ersetzt werden. Das `onpick`-Ereignis wird in Auswahllisten verwendet und kann hier nicht eingesetzt werden. Der `task`-Tag kann eines von den oben beschrieben Elementen sein.

Hier ist ein Beispiel, das die Verwendung des `onenterforward`-Ereignisses verdeutlicht:

```
<card id="card1" title="Erste Card">
<onevent type="onenterforward">
<go href="#card2"/>
</onevent>
...
</card>

<card id="card2" title="Zweite Card">
...
</card>
```

Beim Anwählen eines Links, der auf das Deck oder die erste Karte zeigt, wird sofort die zweite Karte angesprungen. Dieses Beispiel kann für den Fall des `go`-Tasks folgendermaßen vereinfacht werden:

```
<card id="card1" title="Erste Card" onenterforward="#card2">
...
</card>

<card id="card2" title="Zweite Card">
...
</card>
```

Diese Vereinfachung funktioniert auch mit dem `ontimer`- und dem `onenterbackward`-Ereignis in Verbindung mit dem `go`-Task. Das `onpick`-Ereignis nimmt eine Sonderstellung ein, da es sich nicht auf die Ereignisse von Karten und Decks, sondern auf die Auswahllisten bezieht, die durch den `option`-Tag definiert werden. Das `onpick`-Ereignis wird also später im Zusammenhang mit den Auswahllisten im Kapitel 3.2.4.7 behandelt.

3.2.4.6 Templates

Templates bieten die Möglichkeit, Sofkeys oder Ereignisse global für alle Karten eines Decks zu definieren. Es können innerhalb einer Karte zusätzliche Softkeys und Ereignisse definiert oder die im Template definierten Elemente überschrieben werden. So kann z.B. der `noop`-Task dazu verwendet werden, die im Template definierten Tasks auf der Kartenebene auszublenden.
Template wird am Anfang des WML-Decks außerhalb der `card`-Elemente definiert. Das folgende Beispiel zeigt einen Softkey, der für das ganze Deck, also für beide Karten, im `template`-Element initialisiert wird:

```
<wml>
```

```
<template>
<do type="prev" label="Zurück">
<prev/>
</do>
</template>

<card id="card1" title="Erste Card ">
...
</card>

<card id="card2" title="Zweite Card">
...
</card>

</wml>
```

Ein wichtiger Aspekt ist das Ausblenden oder Überschreiben der im Template definierten Elemente. Dieses wird auch als Shadowing bezeichnet. Man unterscheidet zwischen dem Card-Level und dem Deck-Level. Das Card-Level hat eine höhere Priorität. Ein Element, das innerhalb einer Karte definiert wurde, überschreibt die Deck-Level-Definition. Ein Card-Level `onevent`-Element überschreibt ein Deck-Level `onevent`-Element, falls beide dasselbe `type`-Attribut haben. Ein Card-Level `do`-Element überschreibt ein Deck-Level `do`-Element, falls sie den gleichen `name`-Attribute haben.
Nächstes Beispiel zeigt Shadowing, bei dem die im Template initialisierten Elemente ausgeblendet oder überschrieben werden:

```
<wml>
<template>
<do type="prev" label="Zurück" name="sk1">
<prev/>
</do>
<onevent type="onenterforward">
<go href="#card1"/>
</onevent>
</template>
```

Im Template werden ein Zurück-Softkey und ein `onenterforward`-Ereignis, das beim Anspringen der Karte einen Sprung zur ersten Karte auslöst, definiert:

```
<card id="card1" title="Erste Card">
<onevent type="onenterforward">
<noop/>
</onevent>
```

Durch den gleichen Wert des `type`-Attributes wird das im Template definierte Ereignis mit dem `noop`-Task überschrieben. Da der `noop`-Task keine Funktion ausführt, bewirkt man das Ausblenden des Ereignisses.

```
<do type="prev" name="sk1" label="Card 3">
<go href="#card3"/>
</do>
```

Dieses `do`-Element überschreibt durch den gleichen Wert des `name`-Attributes das im Template definierte `do`-Element mit einem Softkey, der sowohl eine andere Beschriftung als auch einen anderen Task hat.

```
<p align="center">
<big><b>Erste Card</b></big>
</p>
</card>
```

```
<card id="card2" title="Zweite Card">
<p align="center">
<big><b> Zweite Card </b></big>
</p>
</card>
```

Die zweite Karte hat keine eigenen Softkey- und Ereignisdefinitionen und übernimmt diese aus dem Template.

```
<card id="card3" title="Dritte Card">

<onevent type="onenterforward">
<noop/>
</onevent>

<onevent type="ontimer">
<go href="#card2"/>
</onevent>
<timer value="30"/>
```

Die dritte Karte überschreibt das Template-Ereignis und definiert ein zusätzliches Timer-Ereignis. Der Wert des Timers ist auf 3 Sekunden eingestellt. Die Sofkey-Defnition wird vom Template übernommen.

```
<p align="center">
<big><b>Dritte Card </b></big>
</p>
</card>

</wml>
```

Zugegebenermaßen würde man in diesem Falle das Ereignis nur in der zweiten Karte definieren und nicht im Template. Das Beispiel ist aber zum Veranschaulichen der Überschreibmechanismen gut geeignet. Die Abbildung 14 zeigt die Navigationsmöglichkeiten, die im obigen Beispiel implementiert sind:

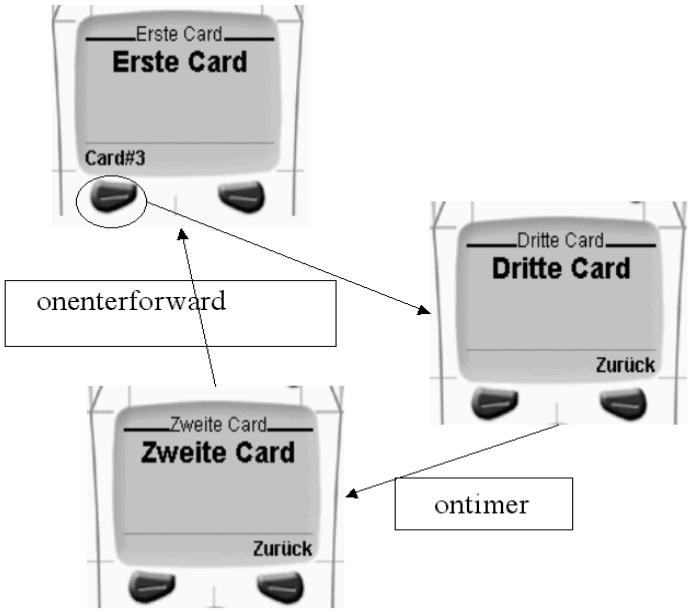


Abbildung 14: Template und Shadowing

WML bietet etwas verschiedene Maskierungsmechanismen für Softkeys und Ereignisse: Die Softkeys werden nicht durch das gleiche `type`-Attribut wie Ereignisse maskiert, sondern durch das Übereinstimmen der `name`-Attribute.
So wird der im Template definierte Softkey vom Typ `prev` durch den Softkey vom Typ `accept` maskiert, da die beiden das gleiche `name`-Attribut haben. Auf der Abbildung 14 ist die Intercard-Navigation des Beispiels gezeigt. Die Karte Nummer drei wird über den Softkey `Card#3` erreicht. Beim Anspringen der Karte Nummer drei wird ein `ontimer`-Ereignis aktiviert und nach drei Sekunden die Karte Nummer zwei angesprungen, bei der sofort ein `onenterforward`-Ereignis aktiviert wird, an das ein `go`-Element mit der ersten Karte als Link angebunden ist.

3.2.4.7 Benutzereingaben

Optionslisten, oder auch Auswahllisten, bilden neben den Texteingabefeldern eine weitere Möglichkeit, von dem Benutzer Informationen zu erhalten.

Um dem Benutzer eine Einfach- oder Mehrfach-Auswahl zu ermöglichen, stellt WML `select`- und `option`-Tags zur Verfügung. Hier ist ein Fragment, das eine Einfach-Auswahl ermöglicht:

```
<select title="EU-Länder" name="var">
<option value="fr">Frankreich</option>
<option value="de">Deutschland</option>
<option value="nl">Niederlande</option>
</select>
```

Dabei wird der Variablen `var` der Wert des jeweiligen `value`-Attribute zugewiesen. Diese Variable kann dann über `$var` abgefragt werden. Man kann nicht nur den explizit zugewiesenen Wert der Option abfragen, sondern auch die Nummer in der Liste des gewählten Eintrages.

```
<select title="Europäische nicht EU-Länder" iname="var">
<option>Schweiz</option>
<option>Ungarn</option>
<option>Lichtenstein</option>
</select>
```

So enthält die Variable `$var` den Wert 1, falls der erste Eintrag ausgewählt wird.
Man kann die Default-Werte setzen, in dem man die Attribute `value` oder `ivalue` im `select`-Tag setzt:

```
<select title="EU-Beitrittskandidaten" iname="var" ivalue="1">
<option>Polen</option>
<option>Slowakei</option>
<option>Tschechien</option>
</select>
```

Mehrfachauswahllisten werden durch Zuweisen des Wertes `true` in dem `multiple`-Attribute definiert:

```
<select title="Skandinavische Staaten " iname="var" multiple="true" >
<option>Dänemark</option>
<option>Finnland</option>
<option>Schweden</option>
</select>
```

Will man mehrere vorselektierte Einträge definieren, so kann man wiederum das `value`- oder `ivalue`-Attribut benutzen und die Werte durch Kommata getrennt als Parameter angeben. Genauso erscheinen die ausgewählten Werte in der Variablen durch Kommata getrennt.
Eine andere Möglichkeit, die Listen zu verwenden, ist der Einsatz des oben erwähnten `onpick`-Ereignisses. Dabei kann aus einer Liste von Links ein Link selektiert und aktiviert werden:

```
<select title="Ehemalige UdSSR Staaten" >
<option onpick="link1.wml">Ukraine</option>
<option onpick="link2.wml">Weißrussland</option>
<option onpick="link3.wml">Moldau</option>
</select>
```

Texteingabefelder werden dazu verwendet, die Eingaben vom Benutzer entgegenzunehmen und den Wert einer Variablen zuzuweisen. Der `input`-Tag muss ein `name`-Attribut erhalten, dessen Wert den Variablennamen enthält. Diese Variable wird mit der Eingabe des Textfeldes initialisiert. Der WML-Codefragment

```
<input name="var">
```

speichert die Benutzereingaben in der Variablen `var`, deren Wert mittels `$var` abgerufen werden kann.
Um die Benutzereingaben zur Fehlervermeidung zu beschränken, stellt WML ein `format`-Attribut zur Verfügung. Der Wert dieses Attributs ist ein regulärer Ausdruck, der auch als Type-Maske bezeichnet wird. Die Type-Maske legt fest, welche Zeichenfolge als Eingabe akzeptiert wird.
Die möglichen Formatierungszeichen sind:

- `A` Großbuchstabe oder Sonderzeichen (ausgenommen Ziffern)
- `a` Kleinbuchstabe oder Sonderzeichen (ausgenommen Ziffern)
- `N` Ziffer
- `X` Großbuchstabe, Ziffer oder Sonderzeichen, wobei das Umschalten auf Kleinschreibung nicht möglich ist.
- `x` Kleinbuchstabe, Ziffer oder Sonderzeichen, wobei das Umschalten auf Großschreibung nicht möglich ist.
- `M` Großbuchstabe, Ziffer oder Sonderzeichen, Umschalten auf Kleinschreibung ist möglich.
- `m` Kleinbuchstabe, Ziffer oder Sonderzeichen, wobei das Umschalten auf Großschreibung möglich ist.
- `*f` beliebige Anzahl von Zeichen, die dem Format `f` entsprechen (`f` kann `A`, `a`, `X`, `x`, `N`, `M`, `m` sein)
- `nf` n Zeichen entsprechend dem Format `f`, z.B. `4N` entspricht vier Ziffern
- `\c` festes Zeichen, das von dem Gerät vorausgefüllt wird.

Beispiele für den Wert des `format`-Attributes und die entsprechende mögliche Eingabe:

```
format="3N\-7N"     040-7663421
format="*A\?3\2a"   ADF?3as
```

Diese Technik kann dazu eingesetzt werden, die Eingabe von einem Text durch eine Ziffernblock-Tastatur der mobilen Geräten zu erleichtern und somit die Fehler gleich bei der Eingabe zu vermeiden. Als Datenformatvalidierung ist dies eine sehr leistungsfähige Technik,

die jedoch wegen mangelnder Unterstützung von Endgeräten dem Zweck der Datenvalidierung nicht dienen kann.
Der `input`-Tag hat einige andere Attribute, die das Verhalten des Texteingabefeldes beeinflussen können:

- Das `type`-Attribut kann die Werte `text` oder `password` haben. Die Default-Einstellung ist `text`. Falls man `password` einstellt, erscheinen die eingegebenen Zeichen durch Asterixe oder ähnliche Zeichen maskiert. Der Wert des Eingabestrings wird nicht beeinflusst.
- Durch das `value`-Attribut kann ein vordefinierter String vorgegeben werden, der im Eingabefeld erscheint.
- Das `emptyok`-Attribut spezifiziert, ob das Feld leer gelassen werden kann. Falls `emtyok="false"` gesetzt ist, wird von dem User Agent ein Mechanismus aktiviert, der weiteres Navigieren durch die Karte verhindert.
- Weitere Tags können die Größe des Eingabefeldes, maximale Länge der Zeichenkette, etc. vorgeben.

3.2.4.8 Textformatierung

Mit Textformatierungstags gibt WML die Möglichkeit, dem Text ein bestimmtes Aussehen zu verleihen. Die Tabelle gibt wieder, welche Möglichkeiten es gibt, den Text zu formatieren.
Es gibt eine Reihe von Textformatierungstags, die die Ausrichtung, das Aussehen und die Größe des Textes beeinflussen. Folgende Tabelle zeigt eine Übersicht der Tags und ihre Bedeutung.

Tag	Bedeutung
`<b>`	Fettschrift
`<big>`	Größerer Schriftgrad als der Standardschriftgrad
`<em>`	Hervorgehobene Schrift
`<i>`	Kursiv
`<small>`	Kleinerer Schriftgrad als der Standardschriftgrad
`<strong>`	Besonders hervorgehobene Schrift
`<u>`	Unterstrichen

Tabelle 7: Textformatierungstags

WML definiert nicht, wie die Textformatierungstags von den Endgeräten interpretiert werden sollen. Die Tags können von verschiedenen Geräten unterschiedlich verarbeitet und sogar ignoriert werden.
Andere Tags, die nicht die Schriftart, sondern die Ausrichtung und den Zeilenumbruch beeinflussen, sind:

- `<br/>` legt einen Zeilenumbruch fest
- `<p>` definiert einen Textabsatz und kann durch das `align`-Attribut die horizontale Ausrichtung des Textes festlegen. Die möglichen Werte sind `left`, `right`, `center`, wobei `left` die Default-Einstellung ist. Durch das `mode`-Attribut kann festgelegt werden, ob ein automatischer Zeilenumbruch erfolgt oder nicht. Falls `mode` auf `nowrap` gesetzt wird, werden die Zeilen nicht umgebrochen. Es hängt vom User Agent ab, wie `mode="nowrap"` verarbeitet wird, z.B. durch das Abschneiden der überschüssigen Zeichen, horizontales Scrollen der Zeile etc.

3.2.4.9 Tabellen

Eine weitere Möglichkeit, WML-Seiten zu gestallten, stellen Tabellen dar. Die Tags, die zum Aufbau einer Tabelle benötigt werden, sind dieselben wie in HTML:

```
<table columns="2" >
<tr><td>11</td><td>12</td></tr>
<tr><td>21</td><td>22</td></tr>
</table>
```

Abbildung 15: Tabelle

Im Unterschied zu den HTML-Tabellen sind einige Beschränkungen hinzunehmen. So ist keine Schachtelung von Tabellen möglich und die Zellen dürfen nicht zusammengefasst werden. Die Zellen der Tabellen dürfen laut WAP-Spezifikation sowohl einen formatierten Text als auch Bilder enthalten. Außerdem kann eine Überschrift durch das `title`-Attribut des `table`-Tags der Tabelle zugewiesen werden.

3.2.4.10 Bilder

In WML stellen Bilder die einzige multimediale Komponente dar. WML unterstützt Bilder im eigenen WBMP-Format. WBMP ist ein Format für Bilder, das nur eine Farbtiefe von einem Bit hat, also schwarz-weiß Bilder unterstützt. Mit Hilfe des `img`-Tags werden Bilder in eine Karte eingebettet:

```
<img src="bild.wbmp" alt="Bild" localsrc="bild1"/>
```

Die wichtigsten Attribute des `img`-Tags sind:

- Das `src`-Attribut enthält eine URL mit dem Dateinamen des Bildes
- Der Wert des `alt`-Attributs wird angezeigt, falls die Anzeige der Bilddatei nicht möglich ist. Dieses Attribut muss immer spezifiziert werden.
- Das `localsrc`-Attribut gibt den Namen eines lokal definierten Bildsymbols an. Falls das Endgerät lokale Bilder unterstützt und das Bild gefunden wird, zeigt das Gerät das Bild an.

Andere Parameter wie `vspace`, `hspace`, `align`, `height`, `width` dienen zur Ausrichtung und Skalierung des Bildes, werden aber nicht von allen Endgeräten unterstützt.
Bei der Verwendung lokaler Bilder muss immer eine Referenz auf ein alternatives Bild vorhanden sein, falls ein Gerät lokale Bilder nicht zur Verfügung stellt.
Mit Hilfe des WAP-Gateways ist es möglich, andere Formate automatisch ins WBMP-Format zu konvertieren. Davon ist jedoch abzuraten, da die automatische Reduzierung der Farbtiefe auf ein Bit die Qualität der Bilder deutlich vermindert.

3.2.4.11 Mechanismen zum Steuern des internen Zustandes des UA

Der Header ist ein optionales Konstrukt von WML. Er wird mit Hilfe des `head`-Elementes nach dem Prolog, aber vor dem `wml`-Tag definiert und enthält Informationen, die beispielsweise die Zugriffskontrolle oder das Cache-Verhalten beeinflussen.
Der `head`-Tag enthält die Informationen, die sich auf das ganze Deck beziehen. Das `head`-Element kann `access`- und `meta`-Tags enthalten:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<head>
<access .../>
<meta .../>
</head>

<wml>
...
```

Mit dem `access`-Element wird die Zugriffskontrolle definiert. Die `domain`- und `path`-Attribute geben an, von welchen Decks aus das aktuelle Deck erreicht werden kann. Das Beispiel verdeutlicht, wie diese Attribute benutzt werden können:

```
<head>
<access domain="tu-harburg.de" path="/cgi"/>
</head>

<wml>
...
```

Das heißt, dass die Decks, die folgende oder analoge URLs haben, auf das aktuelle Deck zugreifen können:

- `http://wap.tu-harburg.de/cgi/main.wml`
- `http://www.tu-harburg.de/cgi/wap/main.wml`
- `http://tu-harburg.de/cgi/test.cgi`

Das `meta`-Element enthält Attribute, die das Cache-Verhalten des User Agents beeinflussen. So kann die Zeitspanne eingestellt werden, in der ein Dokument im Cache-Speicher des User Agents gehalten wird. Durch die ausdrückliche Aussage der WML-Spezifikation, dass die `meta`-Tags vom User Agent ignoriert werden können, ist die Beeinflussung des Cache-Verhaltens sehr schwierig, wenn nicht unmöglich.
Folgendes Beispiel zeigt das Abschalten des Cache-Speichers bei Vorwärts- und Rückwärtsnavigation:

```
<meta http-equiv="Cache-Control" content="no-cache" forua="true"/>
<meta http-equiv="Cache-Control" content="revalidate" forua="true"/>
```

Dabei wird das `http-equiv`-Attribut vom User Agent als HTTP-Header interpretiert. Das content-Attribut gibt den Inhalt des HTTP-Headers an. Falls man die Lebensdauer des Dokumentes im Cache zeitlich begrenzen will, soll man dem `content`-Attribut den Wert "`max-age=xyz`" zuweisen, wobei "`xyz`" den Wert in Sekunden darstellt. Das `forua`-Attribut weist das WAP-Gateway an, diesen Tag bei der Konvertierung in das Binärformat nicht zu entfernen. Ist das Attribut auf "`false`" gesetzt, muss das WAP-Gateway den `meta`-Tag entfernen.

Das `newcontext`-Attribut ist ein Attribut des `card`-Tags. Falls dieses Attribut auf "`true`" gesetzt ist, wird der User Agent beim Vorwärtsnavigieren zu dieser Karte, also infolge der Aktivierung eines Links, reinitialisiert. Die Reinitialisierung des User Agents beinhaltet folgende Schritte:

- Entfernen aller Variablen des aktuellen Browser-Kontextes aus dem Speicher,
- Entfernen aller Einträge aus dem Navigationsverlauf, in dem die zuletzt besuchten Adressen gespeichert wurden und
- Zurücksetzen des internen Zustandes auf einen wohldefinierten Wert.

3.2.5 WMLScript

WMLScript wurde auf Basis von ECMAScript [Areh2000] entwickelt und ist das Gegenstück zu den in der HTML-Welt verwendeten clientseitigen Skriptsprachen wie JavaScript oder VBScript. WMLScript kann für Berechnungen, Datenauswertungen und Generierung von dynamischen und interaktiven Inhalten ohne die Serveranfragen verwendet werden. WMLScript wurde, wie auch WML, als ein Teil der Anwendungsschicht entwickelt, um auf den Clients mit begrenzten Ressourcen, wie Rechenleistung, Speicher, Datendurchsatz und Batterielebensdauer, zu laufen.
Die vom WMLScript zur Verfügung gestellten Arithmetik- und String-Bibliotheken können eingesetzt werden, um die Benutzereingaben zu überprüfen und dann an den Server zu senden, oder ggf. zu korrigieren bzw. den Benutzer zur Eingabe von korrekten Daten aufzufordern.
Dadurch dass WMLScript modular aufgebaut ist, können die Gerätehersteller eigene Bibliotheken entwickeln, die den nativen Zugriff auf geräteeigene Funktionen ermöglichen. Diese Funktionen können folgende sein: Kalender, Adressbuch, GPS-Modul, Blue-Tooth, etc.
Der Unterschied zu den Skriptsprachen wie JavaScript besteht darin, dass WMLScript vom WAP-Gateway in ein Bytecode, das dem Java-Bytecode ähnlich ist, kompiliert wird. Dieser Prozess ist mit dem der Kompilierung eines Quellcodes einer Programmiersprache vergleichbar, in dem dieselben Phasen wie Syntax-, Semantiküberprüfung und Codegenerierung vorhanden sind [Areh2000].
WMLScript bietet alle Konstrukte, die eine Skriptsprache zu bieten hat:

- Variablen mit Variablen-Scope, verschiedenen Typen, wie `integer`, `float`, `boolean`, `string` und Sondertyp `invalid`. Die Typen werden automatisch umgewandelt und müssen nicht explizit angegeben werden.
- Programmablauf wird durch Anweisungen, wie `if`, `for`, `while` und `return` gesteuert.

Es werden folgende Operatoren unterstützt:

- Boolesche Werte "`true`" und "`false`" unterstützen UND-Verknüpfung "`&&`" und ODER-Verknüpfung "`||`".
- Die numerischen Datentypen können die Operatoren "`+`", "`-`", "/" und "`*`" verwenden. Darüber hinaus kann eine Ganzzahldivision "`div`" und Modulo-Berechnung "`mod`" ausgeführt werden.

Die Priorität der Operatoren entspricht der normalen algebraischen Reihenfolge und kann durch Klammerung verändert werden.

WMLScript enthält mehrere Bibliotheken, die verschiedene Bereiche abdecken:

- Die Bibliothek `Lang` enthält Funktionen wie Variablentypumwandlung, einen Zufallszahlengenerator und Funktionen für Integer-Zahlen.
- Die `Float`-Bibliothek enthält Funktionen für Fließkommaoperationen wie Exponenten- und Wurzelberechnung. Es gibt jedoch keine trigonometrischen Funktionen.
- Die Bibliothek `String` enthält Funktionen, mit denen sich Stringmanipulationen durchführen lassen.
- Die `URL`-Bibliothek stellt Funktionen zum Extrahieren von verschiedenen Elementen einer URL zur Verfügung, wie Host-Name, Port, Protokoll, Pfad etc.
- Die `Dialog`-Bibliothek erlaubt dem Entwickler, aus dem WMLScript Meldungen in Form von Dialogboxen anzuzeigen.
- Mit Hilfe der `WMLBrowser`-Bibliothek ist es möglich, Zugriff auf die WML-Variablen zu erhalten und andere WML-Browser-Tasks wie `refresh` oder `prev` auszuführen.

Hier ist ein Beispiel, das zeigt, wie ein Aufruf aus einem WML-Deck erfolgt und wie die Variablen an eine Funktion in einer WMLScript-Datei übergeben werden.
WMLScript-Datei mit dem Namen `script.wmls`:

```
extern function foobar(var1, var2, var3) {
var result=var1 + var2;
WMLBrowser.setVar(var3, result);
WMLBrowser.refresh();
}
```

Die entsprechende WML-Datei, aus der die WMLScript-Funktion aufgerufen wird, muss folgende Elemente enthalten:

```
<wml>
<card>
<onevent type="onenterforward">
<refresh>
<setvar name="param1" value="2"/>
 <setvar name="param2" value="5"/>
</refresh>
</onevent>
<p> $param1+$param2 = $x
<br/><a href="script.wmls#foobar($param1, $param2, 'x')">Berechnen</a>
</p>
</card>
</wml>
```

Die Funktionen, die außerhalb der WMLScript-Datei, sprich Bibliothek, aufgerufen werden sollen, müssen als extern deklariert werden. An die Beispiel-Funktion werden drei Parameter vom WML-Browser übergeben. Diese könnten auch mittels der Funktion `WMLBrowser.getVar()` ausgelesen werden, mit dem Nachteil, dass der Funktion der Name der WML-Variablen zur Entwicklungszeit bekannt sein muss. Da es keine Möglichkeit gibt, die Werte in WMLScript als Rückgabewerte an WML zu übergeben, muss die WML-Variable mit Hilfe der Funktion `WMLBrowser.setVar()` gesetzt werden. Um den generischen Charakter der Funktion zu erhalten, wurde der Name der WML-Variablen, der der Rückgabewert der Funktion zugewiesen wird, als dritter Parameter an die Funktion `foobar` übergeben. Der Aufruf der Funktion erfolgt mit Hilfe des `href`-Attributes, dessen Wert die Referenz auf WMLScript-Datei und die Funktion selbst enthält. Die Parameter, die an die Funktion übergeben werden, werden durch Kommata getrennt. Falls der Wert der Variablen in derselben Karte angezeigt werden soll, wird die Funktion `WMLBrowser.refresh()` aufgerufen.

3.2.6 WTA/WTAI

WTA User Agent ist eine Erweiterung des normalen WAE User Agents oder WAP-Browsers und bietet einige Telephonie-Funktionen. WTA ist in der WAP 1.2 Version spezifiziert, kann jedoch von Endgeräten mit WAP 1.1-Implementierung teilweise verwendet werden, z.B. einige öffentliche Funktionen wie das Absetzen eines Anrufes. Es gibt einige Unterschiede und Gemeinsamkeiten zwischen den beiden User Agents. Sowohl WTA als auch WAE User Agent bedienen sich der WML- und WMLScript-Komponenten. Ein Teil der Telefonie-Funktionen, die in WTA spezifiziert sind, können mittels WTAI (WTA Interface) auch aus WAE User Agent aufgerufen werden. Die Unterschiede bestehen darin, dass WTA User Agent seine Anwendungen von einem Server erhält, der die Kontrolle über das mobile Funktelefonnetz hat, und die abgerufenen Anwendungen dauerhaft im Speicher des Gerätes, auch Repository genannt, ablegt und sie von dort aus nutzt. Um die WTA-Anwendung dauerhaft zu speichern, bedient sich WTA einer Channel-Definition, einer XML-Datei, die Referenzen auf alle nötigen Ressourcen, wie WML-, WMLScript- und andere Dateien, enthält. Der Kreis der Anwendungsanbieter, die WTA-Applikationen anbieten können, beschränkt sich auf die Mobilfunknetzanbieter oder ihre autorisierten Partner. Es ist aus Sicherheitsgründen notwendig, nicht die ganze Palette von Funktionen, die in WTA möglich sind, für alle freizugeben. So werden einige so genannte öffentliche Funktionen über WTAI auch den nicht autorisierten Anbietern zugänglich gemacht.
Hier sind einige Beispiele der Dienste, die mit WTA-Funktionen realisiert werden können:

- Erweiterte Anrufannahme, z.B. Sperren bestimmter Rufnummern, automatische Umleitung von bestimmten Rufnummern auf andere Anschlüsse oder Anrufbeantworter, Senden einer SMS an den Anrufer.
- Voice-Mail-Funktionen, z. B. Benachrichtigung über neu eingegangene Nachrichten, Anzeige einer Liste von Nachrichten mit den Informationen wie die Rufnummer des Absenders, Zeit, eventuelle Textbemerkungen, Selektieren und Abhören der Nachricht oder Anrufen des Absenders.
- Anzeige einer Liste von eingegangenen Nachrichten, wie Fax, Email, SMS oder Voice-Mail und ihre Weiterverarbeitung.

Die WTAI Bibliotheken gehören zu einem dieser drei Typen:

- Netzwerkunabhängige WTAI-Bibliotheken können in allen Mobilfunknetzen bereitgestellt werden. Auf die Bibliotheken kann nur vom WTA User Agent zugegriffen werden, d.h. dass die Dienste, die diese Bibliotheken nutzen, nur von den Netzwerkanbietern bereitgestellt werden können.
- Netzwerkspezifische Bibliotheken stellen die Funktionen, die nur von bestimmten Netzen unterstützt werden, bereit.
- Öffentliche Bibliotheken erlauben den Zugriff über WAE User Agent und können somit von allen Dienstanbietern verwendet werden.

Es gibt folgende WTAI-Bibliotheken:

Bibliothek	Name	Beschreibung	Type
Public WTAI	`wp`	Öffentlich verfügbaren Funktionen	öffentlich
Voice Call Control	`vc`	Steuert das Absetzen und Annehmen der Anrufe	Netzwerkunabhängig (nicht öffentlich)

Network Text	`nt`	Versenden und Empfangen von Textnachrichten	Netzwerkunabhängig (nicht öffentlich)
Phonebook	`pb`	Telefonbuch-Funktionen wie einen Eintrag vornehmen, löschen, von der Seite übernehmen	Netzwerkunabhängig (nicht öffentlich)
Call Logs	`cl`	Anruf Logging, Zugriff auf verschiedene Informationen zu den getätigten Anrufen	Netzwerkunabhängig (nicht öffentlich)
Miscellaneous	`ms`	Verschiedene Funktionen	Netzwerkunabhängig (nicht öffentlich)

Tabelle 8: WTAI-Bibliotheken

Der Aufruf einer WTAI-Funktion aus einer WML-Seite erfolgt wie folgt:

```
<go href="wtai://library/function;parameters"/>
```

Dabei ist "`library`" der Name der Bibliothek, die die Funktion "`function`" enthält. "parameters" sind durch Semikolon getrennte Werte, die an die Funktion übergeben werden.
Die WTAI-Funktionen können genauso innerhalb von WMLScript aufgerufen werden, wie z.B diese Funktion aus der Public WTAI-Bibliothek:

```
Library.functionName("Parameter");
```

Die öffentliche WTAI-Bibliothek `wp` enthält Funktionen zum Behandeln von Anrufen, Senden von DTMF-Tonfolgen und zum Manipulieren von Telefonbucheinträgen.
So kann das Absetzen eines Anrufs aus einer WML-Datei erfolgen:

```
<go href="wtai://wp/mc;766321002"/>
```

Derselbe Funktionsaufruf aus WMLScript:

```
WTAIPublic.makeCall("766321002");
```

Der Anruf kann mit Hilfe der üblichen Benutzerschnittstelle (Gespräch-Beenden-Taste) terminiert werden.
Die Funktion zum Senden von DTMF-Tönen heißt `sd` in WML und `sendDTMF` in WMLScript und wird analog zur Funktion `makeCall` verwendet.
Die letzte Funktion erlaubt das Hinzufügen von Telefonbucheinträgen. Als Parameter werden die Telefonnummern und der Name der Person übergeben:

```
<go href="wtai://wp/ap;766321002;Max Mustermann"/>
```

oder

```
WTAIPublic.addPBEntry("766321002","Max Mustermann");
```

3.2.7 WML-Erweiterungen

Einige Geräte unterstützen Dateiformate, die über die von WAP-Forum definierten Standards hinausgehen. Diese Dateien können in WML-Tags wie der `a`-Tag eingebettet und als Link angesprochen werden.

3.2.7.1 VCF

Visitenkarten-Format [VCARD96] bietet eine Möglichkeit, ohne die WTAI-Bibliotheken auf das Telefonbuch des Endgerätes zuzugreifen. Das VCF-Format wird von verschiedenen Email-Programmen verwendet. Die VCF-Datei ist eine Text-Datei, deren Format folgendermaßen aussieht:

```
begin:vcard
n:Nachname;Vorname
tel;cell:Mobil-Nr
tel;fax:Fax-Nr
tel;work:Telefon-Nr
x-mozilla-html:FALSE
url:Webseite
adr:;;Strasse;Ort;Bundesland;PLZ;Land
version:2.1
email;internet:Email-adresse
fn: Angezeigter Name
end:vcard
```

Die Anbindung in WML erfolgt wie folgt:

```
<a href="visitenkarte.vcf">Visitenkarte</a>
```

Beim Selektieren des Links soll der User Agent die Visitenkarte laden und diese direkt ins Telefonbuch übernehmen. Der WAP-Server muss für den MIME-Typ `text/x-vcard` konfiguriert sein. Um zu detektieren, ob das Endgerät dieses Format unterstützt, kann man den accept-Header, der vom User Agent an den WAP-Server geschickt wird, abfragen und nach der Zeichenkette `text/x-vacrd` suchen.

3.2.7.2 VCS

Das Ansprechen einer Kalender-Datei [VCAL96] ist genauso einfach wie das Ansprechen einer Visitenkarten-Datei:

```
<a href="termin.vcs">Ihr Termin</a>
```

Für die Verwendung dieses Dateityps soll sowohl der MIME-Type `text/x-vCalender` auf dem Server eingerichtet als auch vom User Agent unterstützt werden. Es wird von dem User Agent accept-Header gesendet, der die Zeichenkette `text/x-vCalender` enthält, falls er die Kalender-Dateien unterstützt. Hier ist eine Beispiel-Datei, die einen Termin definiert.

```
BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
CATEGORIES:MEETING
STATUS:TENTATIVE
DTSTART:20010404T183000Z
```

```
DTEND: 20010404T233000Z
SUMMARY: my birthday party
DESCRIPTION: this is the day I’m getting older
CLASS:PRIVATE
END:VEVENT
END:VCALENDAR
```

3.2.7.3 Klingeltöne

Klingeltöne können von einigen Endgeräten, z. B. Ericsson, heruntergeladen werden. Die Vorgehensweise beim Einbetten von den Klingeltondateien in WML ist die gleiche wie bei Visitenkarten- und Kalender-Dateien:

```
<a href="klingelton.mel">Klingelton laden</a>
```

Der MIME-Typ ist `text/x-vmel`.
Endgeräte anderer Hersteller unterstützen diese Methode nicht. Es besteht aber die Möglichkeit, Klingeltöne per SMS oder spezielle WMLScript-Funktionen zu laden.

3.3 Push

Die Push-Technologie wurde in der WAP-Version 1.2 [PUSH99 und Areh2000] spezifiziert. Push ermöglicht die Daten auf den Client zu laden, ohne eine Anfrage an den Server zu senden. Das Push-Framework beinhaltet Protokolle, die notwendig sind, um die Daten vom Initiator-Server zum Client zu senden. Diese Protokolle sind unabhängig vom WAP-Stapel. Der Server, der das Mobilfunknetz und das Internet verbindet, heißt Push Proxy Gateway (PPG). Der Server, der die Daten an den Client sendet, heißt Push Initiator (PI). Für die Verbindungstrecke über das Internet wird das Push Access Protocol (PAP) verwendet. Für die Mobilfunkstrecke wird Push Over-the-Air (OTA) verwendet, das auf dem WSP aufsetzt.

Die Abbildung 16 verdeutlicht das Distributionsmodell des Push-Mechanismus:

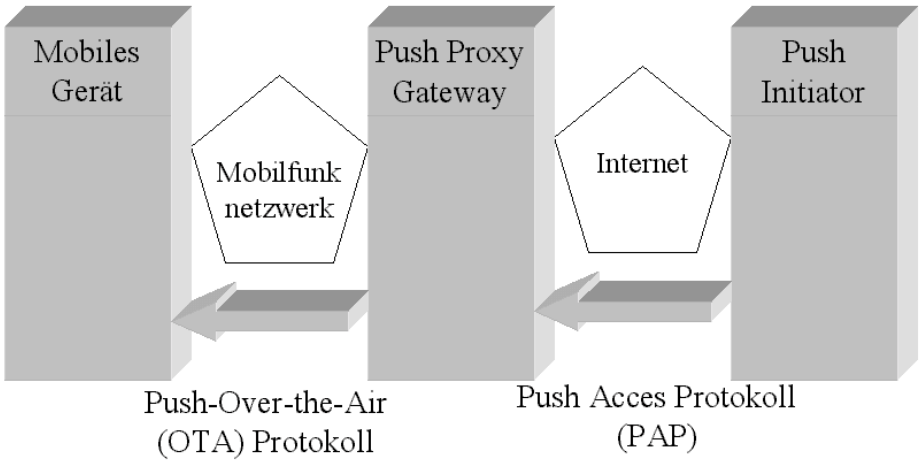


Abbildung 16: Push-Mechanismus

Die Push-Nachrichten, die auf einen Client geladen werden, werden als Nachrichten mit dem MIME-Type `multipart/related` gesendet. Eine Push-Nachricht enthält eine Kontroll-Entität, die in XML-Format definiert ist, und eine Entität mit dem Inhalt, der vom Client angezeigt

oder verarbeitet werden kann. Die Inhalte können vom beliebigen Typ sein, müssen aber vom Client unterstützt werden. Dazu zählen in erster Linie die WAP-Dateiformate, wie WML und WMLScript.
Als Bestätigung wird vom Client eine Push-Antwort gesendet, die dem PI den Status der Lieferung übermittelt, d.h. ob die Push-Nachricht vom UA akzeptiert oder abgelehnt worden oder ein Fehler während der Übehrtragung aufgetretten ist.
Mit Hilfe des Push-Mechanismus sind eine Reihe von WAP-Diensten möglich, wie z. B. folgende Benachrichtigungsdienste:

- Benachrichtigung über Aktienkurse, falls eine Aktie einen bestimmten Wert erreicht hat.
- Benachrichtigung über eingegangene Meldungen wie Emails, Voice-Nachrichten etc.
- Empfang einer elektronischen Zeitung.
- Gebührenabrechnung oder Telefonrechnung, etc.

Bei der Verwendung der Push-Technologie entsteht jedoch ein Problem, denn es können einige PIs an die Endgeräte die Inhalte hoch laden, die vom Benutzer gar nicht erwünscht sind. Es ist durchaus möglich, die Authentisierungsmechanismen und Filterfunktion zu aktivieren, doch je strenger die Restriktionen werden, desto stärker leidet die Benutzbarkeit darunter. Anderseits ist es theoretisch möglich, die Sicherheitsmechanismen umzugehen und z.B. Werbeinhalte zu versenden, wie es heute schon per Email der Fall ist.

3.4 WAP-Sicherheit

WAP bietet einen Mechanismus, der ähnlich der Internet-Sicherungsschicht ist. WTLS basiert auf TLS und gilt bei entsprechend starker Verschlüsselung als sicher. Doch die Problematik ist folgende:
Dadurch, dass WTLS die Funkübertragungsstrecke und TLS die Internetübertragungstrecke unabhängig von einander sichern, liegen die Inhalte auf dem WAP-Gateway praktisch auch für längere Zeit zu Cache-Zwecken unverschlüsselt. Außerdem müssen die Daten dem WAP-Gateway unverschlüsselt vorliegen, um die notwendigen Konvertierungen durchzuführen.
Als Drittpartei hat das WAP-Gateway zugriff auf nicht verschlüsselte Daten, was ein Sicherheitsrisiko darstellt.

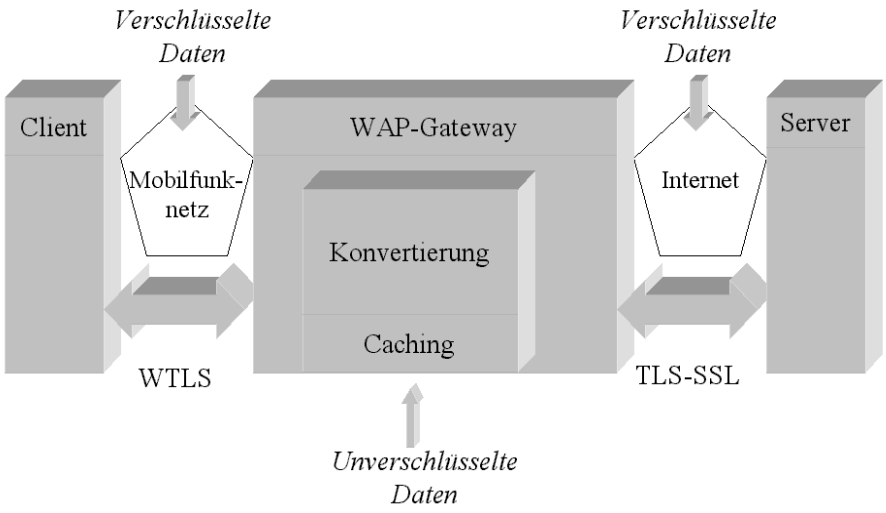


Abbildung 17: Das WAP-Gateway als Sicherheitsrisiko

Deswegen müssen viele Unternehmen, die sicherheitssensitive Transaktionen via WAP durchführen, ihr eigenes WAP-Gateway aufsetzen und den Server entsprechend vom Internet abgrenzen.
Alternativ dazu können die vertraulichen Inhalte auf Serverseite verschlüsselt, in den `setvar`-Tag als Wert des `value`-Attributes substituiert und von einem WMLScript auf dem Client entschlüsselt werden, so dass die Verschlüsselung für das WAP-Gateway transparent bleibt. Die Verschlüsselung von Benutzereingaben oder anderen auf dem Client generierten vertraulichen Informationen ist ebenfals möglich. Die Informationen können mit einem WMLScript auf dem Client verschlüsselt und auf dem Server entschlüsselt werden.

Ein anderes Sicherheitsrisiko besteht darin, dass auf die Variablen einer WML-Seite durch die gleichnamigen Variablen einer anderen WML-Seite zugegriffen werden kann. Das passiert dann, wenn eine WML-Seite Links enthält, die auf die Seiten eines fremden Servers zeigen. Bei dem Aktivieren des Links erfolgt kein Kontextwechsel, d.h. die Variablen werden nicht aus dem Speicher entfernt. Durch die Verwendung von gleichnamigen Variablen können z.B. Passwörter und andere Informationen ausspioniert werden.

Hier ist ein Beispiel, das die Abfrage der gleichnamigen Variablen `$passwort` vom Fremdserver verdeutlicht:

```
...
<card id="deck1_card1">
<input name="passwort">
...
<card id="deck2_card2">
<a href="http://fremdserver/main.wml">Fremder Server</a>
...
```

Die `main.wml` Datei auf dem Fremdserver:

```
...
<card onenterforward="lese_password.wml?$passwort">
</card>
...
```

Deswegen sollten alle sicherheitsrelevanten Variablen vor dem Aktivieren der Links zum fremden Server entweder überschrieben oder gelöscht werden.
Die beiden Beispiele verdeutlichen die Lösungen des Problems:

1) Entfernen aller Variablen aus dem Speicher:

```
...
<card newcontext="true">
<a href="http://fremdserver/main.wml">Fremder Server</a>
...
```

2) Setzen bestimmter Variablen auf nicht relevante Werte:

```
...
<go href="http://fremdserver/main.wml">
<setvar name="passwort" value=""/>
</go>
...
```

3.5 User-Agent-Profile

Die WAP-Version 1.2 verwendet RDF [UAPROF und RDF2000], das Resource Definition Format, zum Beschreiben der Geräteeigenschaften. RDF ist eine XML-Anwendung und dient zur Beschreibung von verschiedenen Internet-Ressourcen. Das WAP-Forum hat RDF adoptiert und an WAP angepasst. Es wird als UAProf bezeichnet. Eine WML-Datei kann beim Herunterladen eine Referenz auf die UAProf-Datei liefern. Der WAP-Server kann UAProf vom Hersteller-Server herunterladen. Die UAProf-Datei liefert Informationen über die Eigenschaften des Endgerätes, wie Hersteller, Art des Gerätes, Modell, Hardware- und Betriebssystemversion, Micro-Browser-Version, unterstützte WAP-Version und Dateiformate, Grafikfähigkeit, etc. Es können aber auch die aktuellen Einstellungen des Gerätes, wie Sound ein oder aus, Bilder ein oder aus, Skriptunterstützung ein oder aus, die z.B. während einer Sitzung geändert werden können, enthalten sein.
Die WML-Spezifikation von UAProf ist noch nicht vollständig abgeschlossen und kann sich bei der nächsten Version des Standards ändern.

Das UAProf-Format sieht vier Abschnitte vor, die jeweils folgende Eigenschaften des Clients beschreiben:

- `HardwarePlattform` enthält alle Hardware relevanten Charakteristika des Endgerätes. Dazu gehören die Informationen über Typ, Model, Displaygröße, Ein- und Ausgabe Methoden der Benutzerschnittstelle, etc.
- `SoftwarePlattform` beschreibt die Eigenschaften der eingesetzten OS-Umgebung, wie Betriebssystem, Video- und Audio-Unterstützung, benutzte Sprache.
- `NetworkCharacteristics` enthält Informationen über das eingesetzte Netzwerk, wie Bearer-Typ und Bandbreite.
- `WAPCharacteristics` stellt die Informationen über WAP-Unterstützung, wie WML, WMLScript, WTA, etc.

Hier ist ein Beispiel des Abschnitts einer UAProf-Datei, die die WAP-Eigenschaften eines imaginären Gerätes enthält [UAPROF]:

```
<rdf:Description>
<prf:WAPVersion>1.1</prf: WAPVersion>
<prf:WMLDeckSize>1400</prf:WMLDeckSize>
<prf:WapDeviceClass>A</prf:WapDeviceClass>
<prf:WapPushMsgSize>1400 octets</prf:WapPushMsgSize>
<prf:WapPushMsgPriority>all</prf:WapPushMsgPriority>
<prf:WtaVersion>1.0</prf:WtaVersion>
<prf:WmlScriptVersion>1.1</prf:WmlScriptVersion>
<prf:WmlScriptLibraries>
<rdf:Bag>
<rdf:li>Float</rdf:li>
<rdf:li>Dialogs</rdf:li>
<rdf:li>URL</rdf:li>
</rdf:Bag>
</prf:WmlScriptLibraries>
<prf:WtaiLibraries>
<rdf:Bag>
<rdf:li>WTAVoiceCall</rdf:li>
<rdf:li>WTAIS136</rdf:li>
</rdf:Bag>
</prf:WtaiLibraries>
<prf:WmlVersion>
<rdf:Bag>
<rdf:li>1.0</rdf:li>
<rdf:li>1.1</rdf:li>
```

```
</rdf:Bag>
</prf:WmlVersion>
```

Die Tabelle enthält einige Eigenschaften aus dem obigen UAProf:

Attribut	Beschreibung	Mögliche Werte
WAPVersion	Unterstützte WAP-Version	1.0, 1.1, 1.2
WMLDeckSize	Maximale Deckgröße in Bytes	z.B. 1400
WapDeviceClass	WAP-Geräteklasse, der das Enfgerät angehört. Die Geräte sind nach Typ, Rechenleistung und Eingabemöglichkeiten aufgeteil	A, B, C
WapPushMsgSize	Maximale Größe einer Push-Nachricht	z.B. 1400

Tabelle 9: Beschreibung einigen Eigenschaften des UAProf-Fragmentes

Genauso wie die WML-Dateien werden die UAProf-Datein für die Übertragung über die Mobilfunkstrecke in ein WXML-Format konvertiert und können zum Cache-Zweck auf dem WAP-Gateway gespeichert werden.

3.6 Diskussion des Standards aus der Sicht des Anwendungsentwicklers

3.6.1 Vorteile

Das WAP-Forum hat sich bei der Entwicklung des Standards bemüht, auf den bestehenden Strukturen und Standards des Internets aufzubauen. So wurden viele Elemente von HTML, insbesondere die Verwendung des URI-Konzepts, um WML-Seiten oder andere Ressourcen wie Bilder, WMLScript-Dateien oder andere Dateien zu referenzieren, in WML übernommen. Durch das URI-Konzept ist eine einfache Anbindung der gerätespezifischen Funktionen sowohl in WML als auch in WMLScript (z.B. Kalender, Visitenkarten, etc.) möglich. D.h. der Anwendungsentwickler referenziert die entsprechende Datei und der User-Agent übernimmt nach der Aktivierung des Links die weitere Verarbeitung des Inhaltes dieser Dateien. Es wird z.B. beim Abrufen einer Kalenderdatei ein entsprechender Eintrag im Terminplaner des mobilen Telefons vorgenommen.
Von der Semantik her ist WML auch sehr stark an HTML angelehnt, d.h. es gibt viele Elemente in WML, die auch in HTML vorhanden sind: Links, Bilder, Tabellen, Eingabefelder und Auswahllisten, etc. Durch neue Elemente, die nur in WML vorhanden sind, wie Ereignisse, Variablen und Softkeys, wird eine Anpassung an die spezifischen Eigenschaften von mobiler Kommunikation, wie knappe Netz-, Rechenkapazitäten und beschränkte Benutzerschnittstellen, durchgeführt. So sind durch diese Elemente Zustandsverwaltung und Verarbeitung im gegebenen Rahmen möglich.
Um komplexere Datenverarbeitung zu realisieren, wird WMLScript eingesetzt. WMLScript ist sehr stark an JavaScript angelehnt und enthält die gleichen Sprachkonstrukte.
WAP verwendet unter anderem dasselbe Distributionsmodell wie das Web. Die WML-Dateien, WMLScript-Dateien, Bilder, etc. werden im Internet mit Hilfe eines normalen Web-Servers zur Verfügung gestellt und durch HTTP-Anfragen zugänglich gemacht. Die Verbindungstrecke WAP-Gateway-WAP-Server wird über TCP/IP-Netze, das Internet, überbrückt.

Das WAP-Forum hat WML als eine XML-Applikation definiert. Dadurch soll ermöglicht werden, die WML-Seiten mit Hilfe von XSLT dynamisch zu generieren oder fertige WML-Seiten weiter zu verarbeiten. Außerdem werden die Probleme mit den Zeichensätzen durch flexible Zeichensatzcodierung in XML vermieden. Dazu lässt sich der Funktionsumfang der Sprache erweitern, um neue Funktionen von Geräteherstellern zu integrieren.

Wie vorher schon erwähnt wurde, bedient sich der WAP-Standard zur Übertragung über die Landstrecke der bewährten Internet-Strukturen. Um die Übertragung über die Funkstrecke möglichst effizient zu gestalten, werden die WML- und WMLScript-Dateien von dem WAP-Gateway konvertiert. WML-Dateien werden in ein kompaktes Byteformat, das dem WXML Format entspricht, umgewandelt. Dabei werden alle in WML definierten Elemente jeweils in einem Byte kodiert. Die Inhalte, die von dem Benutzer stammen, wie Variablennamen, Texte, Labels und Titel, werden als Text, also unkodiert, über die Funkstrecke übertragen. Falls es nötig ist, wird eine Konvertierung in einen von dem Client unterstützten Zeichensatz durchgeführt.
WMLScript Dateien werden in ein dem Java-Bytecode ähnliches Format kompiliert und verbrauchen somit weniger Bandbreite. Alternativ kann die Konvertierung der WML- und WMLScript-Dateien von dem Origin-Server durchgeführt werden, der die WAP-Anwendung bereitstellt. Dadurch wird das WAP-Gateway entlastet und die Übertragung über die Landstrecke etwas beschleunigt.

Durch die Unabhängigkeit vom Typ der Mobilfunknetze (Schichtenmodell) können WAP-Anwendungen von allen Netzen aus verwendet werden, d.h. die Zielgruppe ist nicht auf Kunden bestimmter Netze beschränkt und die WAP-Anwendungen können international genutzt werden.

3.6.2 Nachteile

Einige Nachteile sind in der Natur des mobilen Internets, wie geringe Bandbreite und begrenzte Ein- und Ausgabemöglichkeiten der Endgeräte, die anderen sind im WAP-Standard begründet. Hier ist eine Übersicht der negativen Eigenschaften des WAP-Standards:

- Man muss die WML-Seiten an jedes Gerät anpassen, um den verschiedenen Implementierungen des WAP-Standards, den herstellereigenen Erweiterungen und der Benutzbarkeit gerecht zu werden.
- WML ist nicht mächtig genug, um das gewünschte Layout zu erzielen. WAP ist eine logische Mark-Up-Sprache, die keine Mechanismen enthält, um die sichtbaren Elemente genau auf dem Display zu platzieren. Außerdem würden die kleinen Displaygrößen keine gewünschten Ergebnisse zulassen, da die grafischen Elemente, wie z.B. lange Texte oder große Bilder, größenmäßig der Displaygröße nahe kommen oder diese sogar übersteigen.
- Alle Geräte haben unterschiedliche Benutzerschnittstellenelemente, deswegen ist es nicht möglich bei dem Entwurf einer Anwendung vorherzusehen, wie sie auf einem neuen Gerät aussehen bzw. bedient werden.
- Eine gute Benutzbarkeit ist schwer zu erreichen, da der WAP-Standard keine Implementierungsdetails vorschreibt, an die sich die Gerätehersteller halten müssen, bzw. es gibt kein Mechanismus, der die Eigenschaften der Benutzerschnittstellen-implementierung und anderen Geräteeigenschaften beschreibt, die für die Anwendungsentwicklung notwendig sind.

- Der Standard definiert sehr oft Elemente und verweist den Anwendungsentwickler darauf, dass die Endgeräte diese Elemente nicht unterstützen müssen.
- Viele Hersteller implementieren die Minimallösung, also nur die Elemente, die in der WAP-Spezifikation als Pflicht definiert sind.
- Man weiß nicht genau, ob und wie bestimmte Elemente implementiert sind. Die dadurch entstandene Situation ist der Situation ähnlich, die zwischen Netscape und IE herrscht, doch sie ist viel komplizierter, da sehr viele Geräte auf dem Markt vorhanden sind.
- Das Cache-Verhalten der Geräte ist unbefriedigend, da die Steuerung des Cache-Speichers (Ein-/Ausschalten) nicht auf allen Geräten zuverlässig funktioniert.
- Das WAP-Gateway limitiert maximale Deckgröße. Außerdem kann das Verhalten des WAP-Gateways nicht von der WAP-Anwendung aus beeinflusst werden.

3.6.3 Die Formulierung der WAP-Spezifikation und ihr Einfluss auf die Implementierung

Die vom WAP-Forum entwickelte Standards und ihre Formulierung [WML2000] sind sehr allgemein gehalten. So enthält die Spezifikation von WML bei der Definition vieler Tags folgende Formulierung:

The useragent should do the best effort to...

Für die Gerätehersteller bedeuten solche Formulierungen von Elementen, dass die definierten Tags oder deren Attribute vom Gerät einfach ignoriert werden dürfen. Dies vermindert den Entwicklungsaufwand und erlaubt die Verfeinerung von Funktionen, in dem die späteren Modelle einfach auf Basis der Vorgängermodelle entwickelt und ergänzt werden. Die Offenheit des Standards erlaubt es den Herstellern, durch eigene erweiterte DTDs, sich von der Masse durch spezielle Funktionen abzuheben.
Die Spezifikation legt nicht fest, wie z.B. die Softkeys aussehen oder sich verhalten sollen. So ist der Hersteller frei in der Wahl der Gestaltung der Softkeys. Es können folgende Arten von Softkeys verwendet werden:

- fest verdrahtete Knöpfe mit aufgedruckter Beschriftung
- freiprogrammierbare Knöpfe unterhalb eines Displays, der die von dem Anwendungsentwickler bestimmte Überschrift (label) anzeigt und von ihm belegte Funktion ausführt
- Softkeys, die auf Stimmenkommandos reagieren.

Für den Anwendungsentwickler bedeutet es, dass er nicht vorhersehen kann, ob die Funktion, auf die er gerade aufbaut, auch tatsächlich von dem Endgerät richtig verarbeitet wird. Die Formulierungen der Spezifikation [WML2000] sagen nämlich aus:

Athors must not assume that a user agent implements ...

3.7 Alternativen zu WAP (Dienste und Technologien)

3.7.1 HDML

HDML (Handheld Device Markup Language) ist eine von Phone.com entwickelte Sprache für die Mobilgeräte [Risch2000], die sehr stark im US-Markt vertreten ist. Außerdem sind die meisten datenfähigen schnurlose Telefone mit HDML-Browsern ausgestattet. Alle WAP-fähigen Geräte, die mit dem Phone.com Browser arbeiten, können auch HDML-Dateien anzeigen.
Die HDML-Syntax basiert auf HTML und wurde als Alternative für mobile Geräte entwickelt, um die geringe Bandbreite und schwache Rechenleistung der Geräte zu kompensieren.
Durch die starke Akzeptanz von HDML wurde es als Vorbild für WML genommen (Phone.com ist auch Mitglied des WAP-Forums). So unterscheiden sich die beiden Sprachen nur durch die Syntax; von der Semantik her beruhen sie auf einem identischen Konzept (Decks und Karten, Events, Variablen). Die Anbindung des Mobilgerätes ans Internet erfolgt analog zu WAP über ein Up.Link-Gateway, ein Phone.com eigenes HDML-Gateway, das auch als WAP-Gateway fungieren kann. Der Unterschied liegt darin, dass HDML keine clientseitige Scriptsprache, wie WMLScript, kennt und ausschließlich auf serverseitige Verarbeitung angewiesen ist. Der Vorteil, den HDML mit sich bringt, ist für den Entwickler eines HDML basierten Dienstes gegenüber einem WAP-Dienst folgender: die HDML-Geräte weisen eine höhere Ähnlichkeit zueinander auf, als die WAP-fähigen Geräte, so dass der Dienste-Entwickler sich weniger um die Geräteunabhängigkeit kümmern muss.

3.7.2 iMode

iMode von DoCoMo ist mit über 15 Mio. Teilnehmer in Japan ein großer Erfolg [iMode2000]. iMode stellt einen Dienst dar, dabei ist der Schwerpunkt des Dienstes nicht die Technologie an sich, sondern der Inhalt. Auf der technischen Seite wird eine abgespeckte Version von HTML, cHTML, verwendet. So müssen die Teilnehmer nicht auf die animierte Farbgrafiken und andere Vorteile von HTML verzichten.
Die Übertragungsrate des verwendeten paketorientierten Übertragungsstandards 2G-Mobilfunksystem (PDC) beträgt 9,6 KBit/s, im Frühjahr 2001 wird sie auf bis zu 284KBit/s Downstream und 64 KBit/s Upstream ausgebaut. Dadurch, dass die Teilnehmer immer online sind, entfällt das lästige Einwählen. Die niedrigen Gebühren, ein Micropayment-System via Telefonrechnung und konsequente Auswahl von Partnern, die die Inhalte bereitstellen, tragen dazu bei, eine hohe Akzeptanz und somit hohe Teilnehmerzahlen zu erzielen, ohne neue Standards einzuführen.

Falls man die Inhalte oder Dienste für mobile Geräte mit Hilfe von HTML bereitstellen möchte, muss man beim Design dieselben Kriterien beachten wie bei WAP und HDML. Es geht um eine Datenübertragung über ein Medium mit einer geringen Bandbreite, hohen Latenzzeit und um die Geräte, die keine hohe Rechenleistung und kleine Bildschirme haben. Das bedeutet, dass der Dienstentwickler selbst entscheiden muss, welche multimediale Komponenten er benutzen und worauf er verzichten soll.

3.7.3 Channels

Eine ganz andere Klasse von Browsern bilden die synchronisierenden Browser [Risch2000], die mit Channels arbeiten. Bei einem Channel handelt es sich um eine Gruppe von zusammenhängenden Seiten. Diese Seiten werden einmal heruntergeladen und können dann offline betrachtet werden. Der Vorteil von Channles ist der schnelle Zugriff auf die Informationen, da sie lokal vorliegen. Der Nachteil ist, dass keine wirklich interaktiven Dienste möglich sind.
AvantGo ist ein Vertreter von Channels. AvantGo ist ein Service, bei dem Inhalte kostenlos abonniert werden können. Der Dienst steht sowohl den Palm als auch den Windows CE Benutzern zur Verfügung.
Der Dienst besteht aus drei logischen Komponenten:

- Server
- Desktop des Benutzers
- Mobilgerät des Benutzers mit AvantGo-Browser

Der Server stellt die Inhalte bereit, die mit Hilfe der auf dem Desktop installierten Anwendung aboniert werden. Das Mobilgerät kann dann direkt mit dem Server über eine drahtlose bzw. Festnetzverbindung oder über die Anwendung via HotSync synchronisiert werden.
Microsoft Mobile Channels ist eine andere Technologie, deren Distributionsmodel ähnlich der von AvantGo ist. Dieser Dienst kann nur von Windows CE Besitzern benutzt werden.
Es werden folgende Komponenten eingesetzt:

- Server, der Web-Inhalte enthält. Die Web-Inhalte werden durch eine CDF-Datei zu einem Archiv zusammengefasst. Die CDF-Datei enthält alle verwendeten Ressourcen, wie HTML-Seiten, Bilder, etc.
- Micrisoft Internet Explorer zur Verwaltung von Channel-Abonnements
- Ein Modul zur Synchronisierung des Handheld-Gerätes mit dem Desktop
- Ein Channel-Viewer auf dem Handheld-Gerät

3.7.4 Web Clipping Architecture

Eine weitere Technologie, die den Zugriff auf Web-Inhalte für Geräte mit Palm-OS ermöglicht ist Web-Clipping [CLIPP]. Früher war diese Technologie nur für PalmVII-Geräte vorhanden und somit aufgrund der zur europäischen GSM-Netze inkompatiblen Funkfrequenz nur dem nordamerikanischen Markt vorbehalten. Jetzt kann jedes der Palm-Geräte mit OS-Version ab 3.0 mit entsprechender Software ergänzt werden und über ein Funk- oder Festnetzmodem Web-Clipping nutzen.
Web-Clipping basiert auf HTML 3.2. Die Reduktion der Datenmenge, die über den Funkkanal übertragen wird, findet durch Aufteilung der Inhalte in statische, auf dem Handheld gespeicherte, und dynamische, die vom Server geladen werden.
Die statischen Inhalte bilden die so genannte Web-Clipping-Applikation.
Die dynamischen Inhalte werden als Clippings bezeichnet.
Falls man eine Web-Seite speziell für Web-Clipping entwickelt hat kann man die Seite als Palm-tauglich mit Hilfe von `meta`-Tags definieren.

4 Endgerätebeschreibung

Dieses Kapitel stellt den Kern dieser Arbeit dar und beschreibt die Unterschiede in der WAP-Implementierung von fünf mobilen Endgeräten. Dabei wird auf die gerätespezifischen Eigenschaften und Anomalien der Implementierungen des WAP-Standards eingegangen.
Es gibt eine Reihe von verschiedenen WAP-fähigen Gerätetypen. Dazu zählen mobile Telefone, PDAs, mobile Telefone mit Organizer-Funktionen, Browser für PCs und Software-Emulatoren. Diese Geräte sind für verschiedene Zielgruppen entwickelt worden und haben verschiede Software- und Hardware-Ausstattungen. Einige Endgeräte, vorwiegend PDAs, können verschiedene WAP-Browser verwenden, da diese als installierbare Software implementiert sind und sich jederzeit aktualisieren oder durch einen anderen Browser ersetzen lassen. Anders sieht es bei den mobilen Telefonen aus, deren WAP-Browser ein integrierter Bestandteil des Betriebssystems ist, der sich nicht ohne weiteres auf den neuesten Stand bringen lässt. So können einige Implementierungsfehler durch den Endanwender nicht beseitigt werden. Dafür glänzen die mobilen Endgeräte durch die perfekte Zusammenarbeit zwischen dem WAP-Browser und den integrierten Funktionen des Endgerätes, wie z.B. WTAI, Kalender und Telefonbuch. Die Vorzüge beider Gerätegruppen vereinen die mobilen Telefone mit Organizer-Funktion. Außerdem bieten sie eine bessere Benutzerschnittstelle.
Die Browser für PCs und Emulatoren sind im eigentlichen Sinne keine Geräte, aber weisen alle Eigenschaften eines WAP-fähigen Gerätes auf und werden vor allem zum Testen eingesetzt. Da sie über eine Internetverbindung direkt auf die WAP-Inhalte zugreifen, und somit den Flaschenhals der Funkstrecke umgehen können, ermöglichen sie geringere turnaround-Zeiten bei der Entwicklung. Außerdem kann der Einfluss des WAP-Gateways ausgeschaltet werden, indem man die Zugangsoption über HTTP aktiviert. Dies vereinfacht, beschleunigt und verbilligt den Debugging-Prozess. Der Vorteil der software-basierten Browser ist auch die Möglichkeit, sich den Quellcode der erhaltenen WML-Seite, Variablen und interne Zustände, wie History oder Timer-Werte, anschauen zu können, was bei der Entwicklung dynamischer Inhalte sehr hilfreich ist.

4.1 Allgemeine Beschreibung der verwendeten Geräte

In der Arbeit werden folgende Geräte behandelt:

- Siemens C35
- Nokia 7110
- Nokia 6210
- Sony CMD-Z5
- Palm m100

Die Auswahl bildet einen Querschnitt der auf dem Markt befindlichen Geräte. Nokia 7110 ist das meistverbreitete Gerät, wobei Nokia 6210 einen ähnlichen aber technisch weiterentwickelten WAP-Browser hat. Mit Siemens C35 werden alle Geräte der 35er Serie, also M35 und S35, abgedeckt, da alle Geräte den gleichen Browser haben und die Softkeys gleich implementiert sind. Das Sony-Gerät ist nicht sehr stark verbreitet, ist aber durch die Verwendung des Microsoft-Browsers und durch seine Implementierung von Softkeys ein typischer Browser mit Display-basierten Softkeys, also solcher Sofkeys, die ausschließlich auf dem Display dargestellt werden und keine entsprechenden physikalischen Taste haben.

Das Palm-Gerät ist ein Vertreter der Handheld-Geräte und hat eine für solche Geräte typische große Display-Oberfläche.
Alle Geräte sind laut Herstellerangaben WAP 1.1 fähig und implementieren zum Teil solche Funktionen wie WTAI, Kalender und Telefonbuch.
Das Hauptaugenmerk wird der Unterstützung von WML geschenkt, insbesondere der Vollständigkeit der Implementierung von WML-Elementen, der Realisierung von Sofkeys, dem Cache-Verhalten und den Elemente-Kombinationen, die kritisch sind, d.h. den Elemente-Kombinationen, die in einer WML-Seite nicht richtig dargestellt werden oder bestimmte Endgeräte abstürzen lassen, obwohl laut WML-Spezifikation diese Kombination erlaubt ist.
Außerdem wurden folgende Software-basierte Browser und Emulatoren verwendet:

- Nokia WAP Toolkit 2.0
- Up.SDK 4.0 mit Up.Simulator
- Win-Wap

Die Emulatoren spiegeln die Eigenschaften der Geräte wider. So verhalten sich die Nokia-Geräte annährend wie das Nokia Toolkit 2.0 mit eingestelltem Nokia 7110.
Alle Engeräte, die den Up.Browser von Phone.com verwenden, können recht gut mit dem Up.Simulator getestet werden.
Win-Wap ist ein reiner Windows-Browser und ist sehr gut zum Testen der Programmlogik der Wap-Portale geeignet. Diese Software ist wesentlich fehlertoleranter und verzeiht zum Teil nicht WML-konforme Konstrukte wie das Fehlen der schließenden Tags.
Die Software-Emulatoren und -Browser werden nicht im Detail behandelt.

4.2 Übersicht über verwendete Geräte

4.2.1 Nokia 7110

Das Nokia 7110 Gerät (Abbildung 18) ist das erste funktionsfähige WAP-Handy auf dem Markt.

Abbildung 18: Nokia 7110

Das Endgerät unterstützt WML 1.1, WMLScript 1.1, Bilder im WBMP-Format und einige WTAI-Funktionen. Es kann z.B. eine Telefonnummer von einer WML-Seite aus gewählt werden. Es verarbeitet Decks und Bilder mit einer maximalen Größe von bis zu 1397 Bytes (offizieller Wert von Nokia). Die Displaygröße beträgt 96x65 Bildpunkte. Die maximale Cachegröße ist 40 KByte und die maximale Größe eines Eintrages, der gepuffert werden kann, ist 1536 Bytes. Falls die URL eine Länge von mehr als 255 Bytes hat, wird die Datei nicht gepuffert.
Eigenarten und besondere Merkmale von Nokia 7110 sind:

- Die Textformatierungstags, wie z.B. `b`-, `small`- und `i`-Tags, werden nicht unterstützt.
- Links, Bilder und Eingabefelder werden jeweils in einer neuen Zeile dargestellt.
- Weder vertikale noch horizontale Ausrichtung wird unterstützt. Bilder werden zentriert und Texte linksbündig dargestellt.
- Der Zeilenumbruch ist nicht abschaltbar (wrapping modes, s. Kapitel 4.3.6).
- Bilder können nicht als Links verwendet werden.
- Tabellen werden nicht unterstützt, die Zellen einer definierten Tabelle werden zeilenweise dargestellt.
- Die post-Methode wird unterstützt.
- Die Aufschrift bei der Definition des Zurück-Softkeys darf nicht zugewiesen werden (`label="Zurück"` darf nicht vorhanden sein), die Aufschrift wird der Spracheinstellung des Gerätes entnommen.
- Die maximale Bildgröße sollte nicht 96x44 überschreiten, da sonst das Bild links bzw. unten abgeschnitten wird.
- Es gibt ein vorbelegtes Navigationsrädchen. Man kann damit zwischen den Links navigieren sowie Eingabefelder und Links selektieren.

4.2.2 Nokia 6210

Abbildung 19: Nokia 6210

Das Nokia 6210 Gerät (Abbildung 19) ist ein späteres Modell von Nokia und hat einige wenige Verbesserungen gegenüber dem Nokia 7110 Modell. Die Displaygröße beträgt 96x60 Punkte, wobei der Bereich von 96x41 Punkte für die Inhalt-Anzeige und der Rest ist für die Labels der Softkeys reserviert. Der Cache-Speicher ist 50 KByte groß.

Eigenarten und besondere Merkmale von Nokia 6210 sind:

- Die Textformatierungstags werden nicht unterstützt.
- Links, Bilder und Eingabefelder werden jeweils in einer neuen Zeile dargestellt.
- Horizontale Ausrichtung wird unterstützt.
- Die Bildbreite ist auf 96 Punkte begrenzt.
- Tabellen mit bis zu vier Spalten werden unterstützt.
- Bildanzeige ist benutzerdefiniert abschaltbar.
- Zeilenumbruchfunktion ist benutzerdefiniert abschaltbar.

4.2.3 Siemens C35i

Abbildung 20: Siemens C35i

Die Modelle aus der 35er-Reihe verfügen über identische Software und unterscheiden sich nur in der Displaygröße. Das Display der Modelle C35i und M35i sind 101x54 Pixel, mit bis zu 5 Zeilen, und S35i 101x80 Pixel, mit bis zu 7 Zeilen, groß. Die Geräte arbeiten mit dem Up.Browser 4.x von Phone.com und unterstützen WML 1.1, WMLScript 1.1 und das WBMP-Format. Als zusätzliche Features unterstützen sie eine Reihe von zusätzlichen Formaten und Funktonen, wie Visitenkarten, und WTAI.
Besondere Merkmale sind:

- Die maximale Deckgröße beträgt 2000 Bytes.
- Lokale Bilder werden unterschtützt.
- Die vordefinierte Zurück-Taste (rote Gespräch-Beenden-Taste) kann nicht umdefiniert werden.
- Titel-Leiste kann abgeschaltet werden.
- Verschiedene Textformatierungen werden unterstützt.

- Einträge in einer Select-Liste können mit den Zifferntasten direkt aktiviert werden, ohne vorher selektiert zu werden.
- Karten mit Eingabefelder verhalten sich so, als ob sie aus mehreren Karten bestünden.
- Tabellen werden unterstützt.

4.2.4 Sony CMD-Z5

Sony CMD-Z5 verfügt über ein 96x65 Bildpunkte großes Display. Sony benutzt einen Microsoft-Browser, der zusätzlich auch HTML verarbeiten kann. Die Steuerung erfolgt über ein Jog-Dial-System, das 3 Freiheitsgrade besitzt: Scrollen, Drücken nach vorne, nach hinten und nach rechts.

Abbildung 21: Sony CMD Z-5

Das Gerät hat folgende Merkmale:

- Der "Zurück"-Softkey ist durch zwei vorhandene Hardware-Elemente fest vordefiniert, die c-Taste und Jog Dial nach vorne ziehen. Dieser lässt sich aber auch noch einmal definieren und erscheint dann als Softkey auf dem Display.
- Das Selektieren einer Einfachauswahlliste erfolgt durch Betätigen des Jog-Dial-Rädchen nach hinten.
- Die Softkeys erscheinen dort in der Karte, wo sie definiert wurden, z.B. über der Karte, falls die Definition vor dem Karteninhalt im Quelltext steht. Die Schaltfläche des Softkeys kann aber auch mitten im Text platziert werden. Die im Template definierten Softkeys erscheinen ganz unten und verhalten sich so, als ob sie ganz zum Schluss einer Karte definiert wurden.
- Der Titel wird nicht dargestellt.
- Überschüssige `br`-Tags, also solche, die mehrfach vor einem Link oder Bild definiert sind, werden ignoriert.
- Tabellen werden unterstützt.
- Textformatierungstags werden unterstützt.

4.2.5 Palm m100

Palm m100 ist das derzeit neueste Gerät aus dem Hause 3Com. Es verfügt über ein 2 MByte großen Arbeitsspeicher, ein 160x160 Bildpunkte großes Display und eine IR-Schnittstelle, die für die Verbindung mit einem Mobiltelefon verwendet werden kann. Beim Palm kann beliebige WAP-Browser-Software eingesetzt werden, da die Software im Gegensatz zu den mobilen Funktelefonen installiert und deinstalliert werden kann. Es gibt folgende WAP-Browser:

- AU-System
- WAP-Man
- KBrowser

Es wurde der WAP-Browser von AU-System verwendet, da die anderen Browser aufgrund der geringen Speichergröße des Palm-Gerätes nicht lauffähig waren.

Abbildung 22: Palm m100 mit dem AU-System-Browser

Die Eigenschaften des eingesetzten AU-System-Browsers sind folgende:

- Alle `br`-Tags werden berücksichtig.
- Die Softkeys erscheinen dort in der Karte, wo sie definiert wurden.
- Softkeys sind als Buttons realisiert und werden direkt mit dem Stift aktiviert.
- Die Cache-Größe kann eingestellt werden.
- Tabellen werden unterstützt.
- Textformatierungen werden unterstützt.
- Der Titel wird nicht dargestellt.
- Die Select-Listen werden nicht in einer separaten Karte dargestellt.

4.3 Unterschiede der WAP-Implementierung der Geräte

Dieses Kapitel beschreibt, wie man den WML-Code an verschiedene Geräte anpasst, um ein bestimmtes Ziel, wie z.B. die Anzeige eines Titels oder einer Linkliste, zu erreichen. Dabei werden die besonderen Merkmale in Bezug auf Benutzbarkeit erläutert. Es wird aber auch auf die Fehler in der Implementierung der Geräte hingewiesen.

4.3.1 Titel

Die WML-Spezifikation erlaubt das Verwenden des title-Attributes bei vielen Elementen:

`card`, `select`, `table`, `input`, `option`, `anchor`, `optgroup`, `fieldset`.

Die `optgroup`- und `fieldset`-Elemente werden von keinem der Geräte unterstützt und werden nicht behandelt. Die Elemente `select`, `table`, `input`, `option`, `anchor` werden in der Regel ohne das `title`-Attribut verwendet.
Die Titelanzeige auf einer Karte ist eine wichtige Funktion des `title`-Attributes und wird folgendermaßen aktiviert:

```
<card title="Titel der Card">
...
</card>
```

Die Nokia-Geräte unterstützen das `title`-Attribut. Der Titel erscheint dann in der Titelzeile oberhalb des Karten-Inhalts. Beim Scrollen wird diese Titelzeile nicht mitgescrollt und ist somit immer sichtbar. Sony- und Palm-Browser unterstützen die Anzeige des Titels nicht, d.h. das `title`-Attribut wird ignoriert. Um den Titel dennoch anzeigen zu können, wird der Titel einfach als Text in den Karteninhalt eingesetzt:

```
<card>
<p align="center">
<b>Titel der Card"</b>
</p>
...
</card>
```

Um den Titel grafisch von dem übrigen Text hervorzuheben, wird es zentriert und mit dem `b`-Tag formatiert. Durch diese Vorgehensweise geht die Eigenschaft eines echten Titels, nicht mitzuscrollen, verloren.
Eine Sonderstellung haben die Siemens-Geräte, da die Titelzeile vom Benutzer abgeschaltet werden kann, ohne dass es eine Möglichkeit besteht, dies serverseitig zu detektieren. Als Lösung des Problems kann die Titelzeile als aktiviert oder deaktiviert angenommen werden, oder es kann dem Benutzer die Möglichkeit gegeben werden, die Information darüber, ob Titelleiste ein- oder ausgeschaltet ist, an den Server mittels einer Konfigurationsseite zu übergeben.
Verwendung des `title`-Attributs bei `a`- und `anchor`-Tags hat folgende Auswirkungen:
Bei Siemens wird Titel als Label des Softkeys angezeigt, wenn der Link selektiert wird (ohne `title`-Attribut wird der Text "Link" angezeigt). Nokia, Sony und Palm ignorieren das `title`-Attribut.
Außerdem unterstützt Siemens das `title`-Attribut im `option`-Tag. Der Titel erscheint dann ähnlich wie bei Links als rechter Softkey, wenn die Zeile mit den `option`-Tag-Definition selektiert wird.

Nokia zeigt den Titel des `select`-Tags in der Titelzeile an, während die Einträge Select-Liste berarbeitet werden.
Die Unterstützung von `title`-Attributen anderer Tags erscheint für die Implementierung von interaktiven Anwendungen irrelevant und kann durch Text vor dem entsprechenden Element ersetzt werden.

4.3.2 Softkeys

Die Softkeys werden von den Endgeräten auf verschiedene Art und Weisen realisiert:

- als Schaltflächen auf einem Touchscreen,
- als Schaltflächen auf einem Bildschirm, die durch unterschiedlich realisierte Steuereinrichtungen, wie z.B. jog-dial Rädchen selektiert und aktiviert werden können oder
- als physikalisch vorhandene Knöpfe mit Label, der auf dem Bildschirm über dem Knopf selbst erscheint.

Die Funktion, die mit Hilfe der Softkeys ausgelöst werden kann, kann normalerweise auch frei vorgegeben werden. Es gibt aber Ausnahmen, wie z.B. Tasten, die einen aufgedruckten Label haben, oder Softkeys, deren Funktion nicht umbelegt werden kann.
Falls man eine Vorgabe hat, dass eine Seite bestimmte Softkeys mit einer Funktion und einem Label enthalten soll, dann ist es nicht trivial, dies auf verschiedenen Endgeräten zu realisieren. Der eine Grund dafür ist, dass die Endgeräte verschiedene Anzahl von Softkeys mit unterschiedlichen Restriktionen in Bezug auf Label-Vorgabe, Funktion-Belegung und Mehrfach-Belegung von Softkeys mit dem gleichen Typ haben. Der andere Grund ist eine sehr große Anzahl von Endgeräten, deren Verhalten erst nach dem Testen festgestellt werden kann. Es ist in der WAP-Spezifikation Version 1.1 kein Mechanismus vorgesehen, der dem Entwickler eine Möglichkeit gibt, die Eigenschaften der Benutzerschnittstelle festzustellen.
Die Nokia-Endgeräte haben zwei Softkeys, die auf rechte und linke Tasten abgebildet werden. Der Nokia 7110 hat zusätzlich ein Navigationsrädchen, mit dem man durch das Drehen die Einträge aus einer Liste selektieren oder durch den Inhalt scrollen kann. Durch das Drücken auf das Rädchen kann ein entsprechender Eintrag aktiviert werden. Das Nokia 6210 hat eine entsprechende Einrichtung zum Navigieren, die Auf- und Ab-Tasten.
Das Aktivieren der Einträge erfolgt bei beiden Geräten über die linke Taste.
`do`-Elemente vom Typ "`options`" werden in die Liste von geräteeigenen Optionseinträgen, wie Cache leeren, Lesezeichen, etc, eingereiht. Diese Liste kann über die linke Taste erreicht werden. Es können mehrere `do`-Elemente vom Typ "`options`" initialisiert werden, die in der Initialisierungsreihenfolge in der Liste erscheinen.
`do`-Element von Typ "`prev`" wird auf die rechte Taste abgebildet. Es darf aber kein Label zugewiesen werden, da sonst das `do`-Element so behandelt wird, als wäre er vom Typ "`options`". Das `do`-Element von Typ "`prev`" erscheint aber auf jedem Fall in der Options-Liste ein zweites Mal. Der Label wird abhängig von der im Geräte-Setup eingestellten Sprache "Zurück", "Back", etc. lauten.
Hier ist ein Beispiel, das zeigt, wie die entsprechenden Softkeys eines Nokia 7110-Gerätes belegt werden und aussehen können:

```
<card id="card1" title="Card #1">
<do type="prev"><prev/></do>
<do type="options" label="+Option">
<go href="#card2"/>
</do>
<do type="help" label="+Help">
```

```
<go href="#card2"/>
</do>
<p align="center">
<b>First Card</b>
</p>
</card>
```

Abbildung 23: Softkeys vom Type "help" und "options"

Siemens-Geräte besitzen zwei Display-Tasten, die jeweils links und rechts betätigt werden können. Davon dient die linke Display-Taste zum Navigieren oder Selektieren von Einträgen. Auf die rechte Display-Taste werden die verschiedenen `do`-Typen abgebildet. Diese Abbildung ist hochgradig kontextabhängig, so dass, je nach Inhalt der Karte oder nach Position des Markierbalkens, jeweils einer der definierten Softkeys unmittelbar sichtbar ist. Ein definierter aber nicht unmittelbar sichtbarer Softkey muss durch mehrmaliges Betätigen der rechten Display-Taste rechts sichtbar gemacht werden.
Das `do`-Element vom Typ "`options`" kann über die rechte Display-Taste erreicht werden, falls auf dem Display der Label Menü angezeigt wird. Danach wird eine Liste von allen `do`-Elementen vom Typ "`options`" angezeigt.
Das `do`-Element vom Typ "`prev`" ist mit einer Taste festverdrahtet worden, d.h. weder die Funktion, die nach dem Betätigen dieser Taste ausgeführt wird, noch die Beschriftung lassen sich ändern. Ein Versuch, dieses Verhalten zu überschreiben, wird einfach ignoriert.
Falls man dennoch einen Softkey mit Zurück-Funktion definieren möchte, muss man auf andere dafür geeignete Typen ausweichen. So ein Typ ist z.B. "`accept`". Es sind aber auch viele anderen Typen, wie z.B. "`help`", dazu geeignet.
Der Nachteil dieser Vorgehensweise ist, dass der lokale Kontext, d.h. die Abhängigkeit der Softkey-Belegung von den auf der Karte befindlichen Elementen wie Links, Auswahlmenüs, etc. entscheidet, ob der Softkey sofort zu sehen ist oder ob er erst durch das mehrfache Betätigen der rechten Display-Taste links sichtbar gemacht werden kann. Man kann den Softkey auch durch das Bewegen des Cursors auf einer Stelle, die frei von fokussierbaren Elementen ist, sichtbar machen.
Es wird die Problematik deutlich, dass bei den Endgeräten, die die Sofkeys mit physikalisch vorhandenen Tasten realisieren, das Problem des Mappings auftritt, denn es gibt nur zwei Tasten und eine große Anzahl von `do`-Typen, die auf diese zwei Tasten abgebildet werden können.
Die Sony- und Palm-Browser haben Softkeys, die direkt auf dem Display gezeichnet werden, ähnlich dem Button von den HTML-Browser. Dadurch wird eine "unendliche" Anzahl von Softkeys ermöglicht. Bei solchen Browser ist der Typ der `do`-Elemente nicht relevant, denn alle Typen werden gleich behandelt. Relevant ist jedoch die Reihenfolge der Initialisierung.
Die Elemente, die zuerst definiert werden, erscheinen auch zuerst in der Karte. Dabei gibt es drei Möglichkeiten, die `do`-Elemente im Quellcode zu platzieren:

- unmitellbar nach dem öffnenden `card`-Tag
- unmitellbar vor dem schließenden `card`-Tag
- im `template`-Element

Dabei werden die Softkeys bei den ersten beiden Optionen genau dort visualisiert, wo sie definiert wurden, nämlich am Anfang oder an Ende einer Karte. Die im Template definierten Sofkeys erscheinen ganz unten in der Karte.

Die Verwendung von Softkeys auf eine für alle Geräte uniforme Weise ist aufgrund solcher Vielfalt der Implementierungen nicht möglich. Eine Lösung des Problems ist es, sich auf einige wenige Typen zu beschränken, da viele Geräte maximal zwei Softkeys haben.

4.3.3 Linklisten

Linklisten sind Links, die zu einer Gruppe unter einem Oberbegriff wie z.B. Option zusammengefasst werden. Im einfachsten Fall können die Linklisten als eine Aneinanderreihung von Links definiert werden. Um die Benutzbarkeit von Linklisten auf einigen Geräten zu erhöhen, werden die Linklisten als eine Select-Liste definiert. So ist diese Vorgehensweise bei Siemens-Endgeräten sinnvoll, da die Links mit einer Zifferntaste direkt angewählt werden können. Generell unterstützen alle Geräte mit dem Up.Browser diese Funktion, müssen aber darauf getestet werden, ob das Einsetzen von Select-Listen die Benutzbarkeit erhöht, da die Kontextabhängigkeit (s. Softkey-Beschreibung von Siemens Endgeräten im Kapitel 4.3.2) unterschiedlich realisiert ist. So kann eine Select-Liste bei einem Siemens-Gerät gut funktionieren, dagegen kann die Benutzbarkeit bei einem anderen Up-Browser-basierten Gerät durch unerwünschte Effekte infolge der Kontextabhängigkeit der Softkey-Anzeige verringert werden.
Hier ist eine Gegenüberstellung der beiden Implementierungsmöglichkeiten:

Select-List von Links	einfache Links

`<select>`
`<option onpick="link1">Link1</option>`
`<option onpick="link2">Link2</option>`
`<option onpick="link3">Link3</option>`
`</select>`
	`<a href="link1">Link1</a>`
	`<a href="link2">Link2</a>`
	`<a href="link3">Link3</a>`

Tabelle 10:Select-List vs. Link

Außerdem darf die Select-List-Lösung nicht für Nokia-Endgeräte eingesetzt werden, da aufgrund des Fehlers im Browser der erste bzw. vorselektierte Link nicht aktiviert werden kann.

4.3.4 Texteingabe (`format`-Parameter)

Alle Endgeräte unterstützen das `format`-Attribut des `input`-Elements sehr unterschiedlich. Generell gilt:

- Das Sony-Gerät unterstützt das `format`-Attribut nicht. Die Eingabe fängt mit einem Großbuchstaben an, es können aber auch andere Zeichen durch das Umschalten des Eingabemodus eingegeben werden (das Verhalten entspricht etwa `format="M*m"`, s. unten).
- Der AU-System-Browser des Palm ignoriert das `format`-Attribut ebenfalls.

- Die Nokia-Geräte unterstützen das `format`-Attribut gut. Es gibt aber Ausnahmen, wenn z.B. mehrere Formattypen (z.B. `format="ANANA"`) gemischt werden, kann der Browser die Eingaben nicht richtig validieren.
- Das Siemens-Gerät unterstützt das `format`-Attribut sehr gut und kann fast jede Kombination der Formatierungszeichen richtig verarbeiten.

Folgende Matrix gibt einen Überblick darüber, welche Formatzeichenkombinationen bei welchen Endgeräten funktionieren:

Gerät / Format	Sony	Siemens	Nokia	Palm
`Aaaa*a`	erster Buchstabe groß, jedoch auf klein umschaltbar	min. 4 Buchstaben, erster Buchstabe groß, Punktnationszeichen zugelassen	min. 4 Buchstaben, erster Buchstabe groß	beliebig
`NNN*N`	erster Buchstabe groß, jedoch auf klein umschaltbar	min. 3 Ziffern	beliebig viele Ziffern	beliebig
`XXxx`	erster Buchstabe groß, jedoch auf klein umschaltbar	Zwei große, zwei kleine Buchstaben	nur Kleinbuchstaben	beliebig
`mm\_MM`	erster Buchstabe groß, jedoch auf klein umschaltbar	Vier Zeichen durch "_" getrennt	nur Kleinbuchstaben, "_" wird nicht eingefügt	beliebig
`ANAN`	erster Buchstabe groß, jedoch auf klein umschaltbar	Zeichen, Ziffer, Zeichen, Ziffer	nur Kleinbuchstaben	beliebig
`NN\-NN`	erster Buchstabe groß, jedoch auf klein umschaltbar	Zwei Paar Ziffern durch "-" getrennt	nur Kleinbuchstaben, der Wert wird der Variablen nicht zugewiesen	beliebig
`3N`	erster Buchstabe groß, jedoch auf klein umschaltbar	max. 3 Ziffern	Ziffernfolge beliebiger Länge	beliebig

Tabelle 11: Unterstützung des `format`-Attributes

Es gibt keine Möglichkeit das "richtige" Verhalten auf der WML-Ebene zu erzwingen, d.h. das `format`-Attribut kann nicht als Maske für die Daten verwendet werden, um z.B. korrektes Datenformat sicherzustellen. Mit der Hilfe eines WMLScript kann eine bessere Datenvalidierung implementiert werden. Hier ist ein Beispiel, das die Validierung eines Email-Formates durchführt. Das Email-Format soll dem Format `xx*x\@xx*x\.xxx` , also der Kombination von: min. 2 Zeichen, @-Zeichen, min. 2 Zeichen, ".", 2 oder 3 Zeichen, entsprechen.
Hier ist ein Beispiel-Fragment aus der WML-Datei, in der die WMLScript-Funktion aufgerufen wird:

```
<input name="email" emptyok="false">
<a href="email.wmls#validate ('$(email:unescape)')">Email senden</a>
Gültige Email? $valid
```

Hier ist der Inhalt der Datei `email.wmls`:

```
extern function send(strEmail){

validate(strEmail);
WMLBrowser.setVar("valid", validate(strEmail));
WMLBrowser.refresh();

}

function validate(strEmail) {
var postionOfAt;
var posiotionOfDot;

if(checkForChar(strEmail, "@") !=1){
return "Nein, nicht genau ein @-Zeichen";
}
if(checkForChar(strEmail, ".") <1){
return 'Nein, keine "." gefunden';
}
if(String.find(strEmail, "@") > findLastIndexOf(strEmail, ".")){
return 'Nein,"." vor dem @-Zeichen ';
}
if((findLastIndexOf(strEmail, ".")+3) > String.length(strEmail)){
return 'Nein, nach "." weniger als zwei Zeichen';
}
if((findLastIndexOf(strEmail, ".")+4) < String.length(strEmail)){
return 'Nein, nach "." mehr als drei Zeichen';
}
if(findLastIndexOf(strEmail, ".")-3 < String.find(strEmail, "@")){
return 'Nein, zwischen "." und @ weniger als zwei Zeichen';
}
if(String.find(strEmail, "@") < 2){
return "Nein, vor @ weniger als zwei Zeichen";
}

return "Ja, die Email-Adresse entspricht dem Format xx*x\@xx*x\.xxx ";

}

function checkForChar(str, char){
var numberOfElements;
var numberOfDelimiters;

numberOfElements = String.elements(str, char);
numberOfDelimiters = numberOfElements -1 ;

return numberOfDelimiters;
}

function findLastIndexOf(str, char){
var i;
var tmp;
var length = String.length(str);

for (i =0; i<length;i++){
if(String.charAt(str, i)==char){
tmp=i;
}

}

return tmp;
}
```

Der Aufruf des Skriptes erfolgt über einen Link. Die Variable wird als nicht URL-codierter Wert an die externe Funktion `send()` übergeben, die für den Aufruf der Funktion `validate()` und für das Zurückliefern der Auswertung und Anzeige im Browser verantwortlich ist.

Die Funktion `validate()` führt die eigentliche Überprüfung der Zeichenkette durch und bedient sich weiterer interner Funktionen, die z.B. nach bestimmten Zeichen suchen können und die Anzahl der Vorkommnisse oder die Position des letzten Vorkommnisses zurückliefern.
Der Vorteil von Skripten ist, dass man das Verhalten der Validierung unter Kontrolle hat und nach Bedarf anpassen kann. Der Nachteil ist, dass eine zusätzliche Datei heruntergeladen werden muss, und somit die Auswertung bei der ersten Anwendung mit höheren Zeitaufwand verbunden ist.

4.3.5 Textformatierungen

Die Textformatierungstags werden von keinem der Endgeräte vollständig unterstützt, da nur eine geringe Anzahl von Schriftarten im Gerätespeicher vorhanden ist. Außerdem werden die Textformatierungstags teilweise willkürlich auf die Schriftarten abgebildet, so dass man nur durch das Testen der Geräte diese Zuordnung feststellen kann.
Da eine begrenzte Anzahl von Schriftarten vorhanden ist, ist die Schachtelung von Tags nicht sinnvoll. Außerdem ist das Ergebnis nur nach dem Test ersichtlich. So kann man nicht sagen, welches Ergebnis die Kombination `<i><big>Text</big></i>` liefert. Andere weniger sinnvolle Kombinationen wie `<big><small>Text</small></big>` sollen vermieden werden, obwohl sie laut DTD zugelassen sind.
Folgende Tabelle spiegelt die Abbildung der Tags auf die Schriftarten wider:

Tag / Gerät	`<b>` Bold	`<i>` Italic	`<u>` unterstrichen	`<em>` Emphasis	`<strong>`	`<small>`	`<big>`
Sony	fett	kursiv	unterstrichen	kursiv	fett	normal	groß
Nokia 7110/6210	normal	normal	normal	normal	normal	normal	normal
AU-Browser	fett	normal	unterstrichen	normal	normal	normal	groß
Siemens	fett	kursiv	normal	fett	fett	normal	fett

Tabelle 12: Unterstützung der Textformatierungstags

Wie man sieht, ist der `b`-Tag am besten dafür geeignet, einen Text hervorzuheben, da dieser Tag fast von allen Endgeräten unterstützt wird. Der Tag "`small`" wird von keinem dieser Geräte unterstützt und ist nur für zukünftige Geräte mit besserer Unterstützung sinnvoll.
Obwohl die Unterstützung der Formatierungstags sehr beschränkt ist, ist es jedoch sinnvoll, sie einzusetzen, da bei neuen Endgeräten mit der Verbesserung der Unterstützung zu rechnen ist.

4.3.6 Zeilenumbruch

Das `mode`-Attribute des `p`-Tags bietet die Möglichkeit, die Anzeige von langen Textzeilen zu beeinflussen. Die Standardeinstellung ist `mode="wrap"`, was bedeutet, dass der Browser den Umbruch der Zeilen automatisch vornimmt. Bei Verwendung von `mode="nowrap"` wird die Zeile nicht umgebrochen, was bei verschiedenen Browser zu unterschiedlichen Ergebnissen führen kann. Nokia 7110/6210 und Sony ignorieren das Attribut. Das Nokia 6210 erlaubt das

benutzerdefinierte Aktivieren des Zeilenumbruches für lange Zeilen. Die Siemens-Geräte blenden die Teile der Zeile zeitlich nacheinander ein, so dass das Lesen der ganzen Zeile nur schwer möglich ist. Aufgrund der mangelnden Unterstützung ist die Verwendung von `mode="nowrap"` derzeit nicht sinnvoll.

4.3.7 Auswahllisten

Alle Endgeräte unterstützen die `option`- und `select`-Tags. Die Realisierung ist jedoch unterschiedlich. Die mobilen Telefone öffnen beim Aktivieren einer Select-Liste eine neue Bildschirmseite, die nach dem erfolgten Selektieren vom Benutzer geschlossen werden muss, um zur Karte zurückzukehren.
Die Abbildung 24 veranschaulicht die Implementierung der Select-Liste in einer separaten Karte:

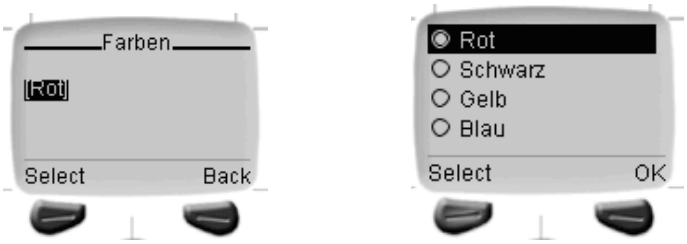


Abbildung 24: Implementierung der Select-Liste in einer separaten Karte

Durch die Aktivierung des "Select"-Softkeys wird die Karte mit den Select-Listeneinträgen angezeigt.
Das Siemens-Endgerät erzwingt das Eintreten in die Select-Liste und die anderen mobilen Telefone erlauben das Überspringen der Liste, ohne die Wahl der Einträge durchzuführen.
Die Up.Browser-basierten Endgeräte, zu denen auch Siemens gehört, unterstützen die direkte Auswahl der Einträge. Die Einträge sind von 1 bis 9 durchnummeriert und können mit den Ziffern-Tasten ausgewählt werden.
Der AU-Browser des Palm stellt die Einträge der Liste auf derselben Bildschirmseite wie die Karte selbst dar.
Die Verwendung der Listen bedarf keiner besonderen Maßnahmen bzgl. der Geräteunabhängigkeit. Die Anzahl der Einträge sollte aus ergonomischen Gründen zehn nicht übersteigen.

4.3.8 Tabellen

Tabellen stellen eine erweiterte Möglichkeit dar, die Daten zu formatieren. Aufgrund der geringen Bildschirmgröße ist es sehr schwierig, ein funktionierendes Konzept für die Darstellung der Daten, die die Größe des Bildschirms übersteigen, zu finden. Dem entsprechend ist die Unterstützung der Darstellung verschiedener Inhalte wie langer Texte, Links und Bilder in einer Tabelle mangelhaft.

Folgende Tabelle gibt einen Überblick über die Unterstützung der in einer Tabelle verwendeten Elemente:

Inhal / Gerät	kurzer Text	Langer Text	Bilder	Links
Sony	ja	Zellen werden zeilenweise dargestellt	OK	OK
Siemens	ja, kein Rahmen	Lange Zeilen werden auf Folgezellen fortgesetzt, Spalten verschoben (`mode="nowrap"`), Teile der Zeile werden zeitlich nacheinander eingeblendet (`mode="wrap"`)	keine Bilder und kein alt. Text	OK
Nokia6210	ja, mit Rahmen	Abgekürzt, nach einer Auswahl der Zelle voll, `mode`-Attribut wird ignoriert	alternativer Text	OK
Nokia7110	Zellen werden zeilenweise dargestellt	-	Zellen werden zeilenweise dargestellt, Bild zentriert	-
AU-Browser	ja, mit Rahmen	-	-	-

Tabelle 13: Tabellenunterstützung

Aus der Tabelle ist es ersichtlich, dass man bei der Entscheidung, Tabellen zu benutzen, sich auf sehr kurze Texte beschränken sollte. Lange Texte sind entweder erst nach dem Aktivieren der Zelle lesbar oder die Tabelle ist nicht mehr als eine Tabelle zu erkennen. Bilder werden nur von einigen Geräten angezeigt, und es wird teilweise auf den alternativen Text ausgewichen oder der `img`-Tag wird ignoriert.

4.3.9 Bilder

Alle hier behandelten Geräte unterstützen Bilder im WBMP-Format. Die Bilder können sowohl im Text als auch mit einigen Einschränkungen in Tabellen, Links und sogar theoretisch als Softkey-Labels verwendet werden. Die Unterstützung der Bilder in Tabellen wird im Kapitel 4.3.8 behandelt. Auf die Verwendung der Bilder in Links wird im Kapitel 4.3.10 eingegangen. Das Nokia 6210 ermöglicht das benutzerdefinierte Abschalten der Bildunterstützung, d.h. anstelle des Bildes wird der alternative Text angezeigt.
Es gibt einige Beschränkungen bzgl. der Bildgröße in Bildpunkten:
Das Nokia 7110 kann die Bilder nur bis zur Größe von 96x44 richtig verarbeiten. Die Breite der Bilder darf beim Nokia 6210 nicht mehr als 96 Bildpunkte betragen, und falls das Bild die Bildschirmhöhe überschreitet, kann gescrollt werden. Bei Sony-Geräten können die Bilder, die Bildschirmgröße übersteigen, nicht dargestellt werden. Außerdem gelten für die Geräte die Dateigrößenbeschränkungen, die beachtet werden müssen.
Das Nokia 7110 hat abhängig von der Software-Version beim Auftreten der Kombination von `setvar`- und `img`-Tags in einer Karte folgendes Verhalten (s. auch Kapitel 5.3.3.5):

- Die Karte wird richtig verarbeitet.
- Es wird eine Fehlermeldung ausgegeben.
- Das Endgerät stürzt ab, so dass ein Neustart des Gerätes notwendig ist.

Man muss also beim Nokia 7110 sicherstellen, dass diese Kombination nicht vorkommt.
Bei der Positionierung der Bilder im Text gibt es auch einige Unterschiede. So werden die Bilder bei Nokia-Geräten immer in einer neuen Zeile positioniert, d.h. die Geräte ergänzen intern einen `br`-Tag, falls dieser nicht vorhanden ist. Um uniformes Aussehen der Bilder auf allen Geräten zu erreichen, sollte also generell vor einem `img`-Tag ein `br`-Tag eingefügt werden.
Die Darstellung von lokalen Bilder ist nur den UP.Browser-basierten Endgeräten, also auch Siemens, vorbehalten. Die Vorteile der lokalen Bilder sind zu einem keine langen Download-Zeiten, zum anderen standardisierte Motive, die auf allen Geräten, die lokale Bilder unterstützen, gleich sind. Hier sind einige Bilder, die von den Up.Browser eingesetzt werden:

`localsrc`-Attribut	Bild	Nummer
`document2`		103
`house`		112
`righthand`		141
`dollar`		139
`phone1`		85

Tabelle 14: Unterstützung der lokalem Bilder

4.3.10 Links

Auch bei Implementierung der Darstellung von Links gibt es gerätespezifische Unterschiede.
So erscheinen die Links auf Nokia-Geräten jeweils in einer neuen Zeile.
Siemens-Geräte stellen die Links nicht unterstrichen wie alle anderen Geräte, sondern in eckigen Klammern dar.
Der WAP-Standard erlaubt die Verwendung von Bilder in den Links, doch nicht alle Geräte unterstützen dies. So kann das Nokia 7110 einen Link als Bilder nicht darstellen und zeigt dafür den alternativen Text an.
Eine andere Situation entsteht, falls ein Siemens-Endgerät eine Kombination aus Link gefolgt von einem Eingabe-Element darstellen muss. Durch diese Kombination ist es nicht möglich, auf den Link zuzugreifen. Abhilfe schafft die Verwendung von `p`-Tags, d.h. die beiden Elemente müssen innerhalb von verschiedenen `p`-Elementen definiert sein, was eine problemlose Navigation zwischen den Elementen sichert.
Eine weitere Problemquelle ist die fehlende Unterstützung der post-Methode bei den Siemens-Endgeräten.
Die Implementierung der get-Methode beschränkt die Übergabe von Parametern durch die Begrenzung der maximalen Länge von URL-Strings.
Nokia-Geräte haben einen Fehler in der Implementierung der relativen Pfadangaben. Beim Vorkommen eines "/" Zeichens in dem Parameterteil des URL-Strings, also nach dem Fragezeichen, kann nicht mehr relativ adressiert werden. D.h die Verwendung eines Abfragestrings der Form:

```
root/test/test_1.wml?a=test/test_x.wml
```

bringt die interne Implementierung der Linkverwaltung durcheinander, so dass darauf folgende Linkaufrufe falsch ausgeführt werden:

```
../ernst/ernst_1.wml
```

Die Pfadangabe muss also relativ zum Root-Verzeichnis des Servers erfolgen, z.B. eine Referenz auf die Datei `http://wap.test.de/root/ernst/ernst_1.wml` muss als `/root/ernst/ernst_1.wml` gemacht werden, und zwar unabhängig von dem Ort der referenzierenden Datei.

4.3.11 Cache-Verhalten

Alle Geräte besitzen einen Cache-Speicher, dessen Verhalten sich mit `meta`-Anweisungen steuern lassen soll. Es gibt aber Ausnahmen wie Siemens-Geräte, deren Cache sich nicht abschalten lässt. D.h. das Laden von dynamischen Seiten kann nicht sichergestellt werden. Um das Laden von dynamischen Seiten jedoch zu ermöglichen, muss an den URL-String eine sich mit jedem Aufruf ändernde Zeichenkette angehängt werden.
Außer Siemens gibt es ein anderes Verhalten aller Geräte, die beim Rückwärtsgehen die Daten nicht aus dem Cache laden, sondern die Seiten von dem Server anfordern, obwohl die Seiten eigentlich aus dem Cache geladen werden sollten. Es gibt keine Möglichkeit diesem Verhalten entgegenzuwirken, außer einer serverseitigen Erkennung des Rückwärtsgehens anhand eines Zählers, der serverseitig gesetzt und von Client bei jeder Anfrage an den Server gesendet wird. Durch den Vergleich des Zählers mit dem erwarteten Wert kann festgestellt werden, ob ein Rücksprung stattgefunden hat, und es kann entsprechend darauf reagiert werden.
Um dasselbe Verhalten des Siemens-Gerätes, d.h. das Laden der Seite vom Server beim Rückwärtsgehen, also kein Caching, zu erzwingen, muss ein `meta`-Tag innerhalb des `head`-Elements vorhanden sein.

```
<meta http-equiv="Cache-Control" content="revalidate" forua="true"/>
```

4.3.12 Deckgrößenbeschränkung

Fast alle Geräte haben einen sehr knapp bemessenen Speicher. Eine Folge davon ist die Beschränkung der Deckgröße maximal zwei KByte. Bei dynamischen Inhalten ist es zum Teil nicht möglich, die Menge der Daten, die in einem WML-Deck dargestellt werden, vorher zu bestimmen. Es muss also einen serverseitigen Mechanismus geben, der die dynamischen Inhalte auf mehrere Decks aufteilt.
Die Deckgrößenbeschränkung hängt aber nicht nur von den Endgeräten selbst, sondern auch von dem eingesetzten WAP-Gateways ab. Das UP.Link-Gateway hat eine Beschränkung der Deckgröße auf 2000 Bytes.

4.4 Ergebnisse der Problemanalyse

Die Analyse der WAP-Implementierung und der WML-Implementierung im Speziellen zeigt, dass es viele Defizite sowohl in der Spezifikation als auch in der Implementierung dieses

Standards gibt. Zum Teil verursacht die Spezifikation die Mängel in der Implementierung der Elemente, dadurch dass die Elemente als optional spezifiziert werden (s. Kapitel 3.6.3).
Als Folge davon haben viele Geräte eine unvollständige WML-Implementierung:
Textformatierungstags und Tabellen werden nicht von allen Geräte unterstützt, das `title`-Attribut wird von vielen Geräten ignoriert.
Die WML-Spezifikation erlaubt das Aussehen des Textes durch wenige Elemente wie Textformatierungstags und `mode`-Attribut, mit dem sich das Zeilenumbruchverhalten steuern lässt, zu beeinflussen. Es bietet jedoch keine Mechanismen das Layout genau zu gestallten.

Bei einigen Geräten ist der WAP-Standard sogar fehlerhaft implementiert, so führt die Kombination aus `setvar`- und `img`-Tags bei dem Nokia 7110 in einigen Fällen sogar zum Absturz des Gerätes (s. Kapitel 4.3.9), die Select-List-Implementierung der Nokia-Geräte bei dem Einsatz des `onpick`-Ereignisses (s. Kapitel 4.3.3) lässt die Aktivierung des ersten bzw. selektierten Eintrages nicht zu.
Außerdem ist die Steuerung des Cache-Speichers äußerst unzureichend, wie es am Beispiel der Cache-Aktivierung im Zusammenhang mit der Rückwärtsnavigation (s. Kapitel 4.3.11) deutlich wird.

Die Gerätehersteller verfolgen unterschiedlichste Ansätze bei der Softkey-Realisierung und der Realisierung von anderen Elementen, wie Select-Listen. Es muss für jedes Gerät experimentell die optimale Lösung bestimmt werden.
Durch die Vielfallt von verschiedenen Geräte ist es nicht möglich, eine WML-Untermenge zu finden, die als kleinster gemeinsamer Nenner für alle Geräte eingesetzt werden kann.

Aus diesen Punkten ergibt sich die Notwendigkeit zusätzlliche Mechanismen zu implementieren, die diese Defizite kompensieren, insofern es möglich ist. Außerdem sind einige Anpassungen an das WML-Zustandsmodell von Vorteil, die das Variablen-Konzept z.B. für die Sitzungsverwaltung ausnutzt.

Die zu implemenierenden Mechanismen sind folgende:

- Unterstützung von unterschiedlichen WAP-fähigen Clients mit Hinblick auf Skalierbarkeit, d.h einfach Erweiterbarkeit mit neuen Endgeräten, flexible Anpassung an die Eigenschaften der Geräte wie z.B. die Softkey-Implementierung,
- automatische Identifikation und Verwaltung der Geräte,
- serverseitige Kontrolle des Cache-Verhaltens, bei Vorwärts- und Rückwärtsnavigation und
- an WML-angepasste Sitzungsverwaltung
- Kontrolle der Deckgröße

5 Entwurf und Realisierung eines geräteunabhängigen WAP-Portals

Dieses Kapitel geht auf den Funktionsumfang eines WAP-Portals und die Aspekte ein, die bei der Wahl der aus dem Web-Portal portierten Dienste berücksichtigt werden müssen.
Außerdem werden die für die Realiserung der Geräteunabhängigkeit notwendigen Techniken beschrieben. Es werden zwei Standbeine der Lösung erläutert:
Ein Teil der Lösung ist softwaretechnischer Natur und implementiert alle Mechanismen, die die Geräteunabhängigkeit unterstützen. Der andere Teil stellt eine Richtlinie dar, die für den Anwendungsentwickler als eine Vorschrift dient und beschreibt, wie er die WML-Templates gestalten soll, damit die Templates mit dem infoAsset Broker ordnungsgemäß funktionieren.

Diese zwei Standbeine werden durch drei Ansätze miteinander verbunden. Diese Ansätze bestimmen die Art, wie die Geräteunabhängigket erreicht wird und zwar:

- durch die Entwicklung generischer WML-Seiten
- durch die Entwicklung geräteoptimierter WML-Seiten
- durch automatische Generierung geräteoptimierter WML-Seiten

Diese drei Ansätze kommen gleichzeitig, je nach Bedarf zum Einsatz.

5.1 Funktionsumfang des WAP-Portals

Das WAP-Portal soll einige Dienste des Web-Portals (s. Kapitel 2.8.5) enthalten und besteht aus folgenden vier Bereichen (Abbildung 25):

1. Zugangsdienst besteht aus Anmeldung, Registrierung und Gastzugang.
2. Verzeichnisdienst stellt die Verzeichnisbaumstruktur bereit, die sowohl Verzeichnisse als auch die darin gespeicherte Dokumente anzeigt.
3. Sammelmappen ermöglichen die Speicherung und das Ablegen von Links, Dokumenten, Verzeichnissen und anderen Assets.
4. Dokumente (nicht in der Abbildung 25) erlauben das Anzeigen von Zusatzinformationen und erlauben das Sichten von Dokumenten.

Diese Dienste sind an die des Web-Portals angelehnt und stellen einen für die mobilen Geräte angepassten Funktionsumfang dar (Abbildung 25).

`anonymous.wml` ist die Startseite und bietet Links, die es einem Benutzer ermöglicht:

- sich durch die Aktivierung von `anonymous.wml#c2` anzumelden,
- sich durch die Aktivierung von `registration.wml` zu registrieren,
- durch die Aktivierung von `welcome.wml` den Gastzugang zu nutzen, ohne ein registrierter Benutzer zu sein

Die Karte `c2` des Decks `anonymous.wml` enthält Benutzername- und Passwort-Eingabefelder sowie einen Senden-Link, bei dessen Aktivierung die Anmeldung des Benutzers mit den in die Eingabefelder eingetragenen Daten durchgeführt und anschließend auf die Seite `welcome.wml` umgeleitet wird.

Das Deck `registration.wml` enthält die Eingabefelder für die Benutzerinformationen, wie Vor- und Nachname, Anrede, Telefonnummer, Email-Adresse, etc., sowie einen Senden-Link, bei dessen Aktivierung die Registrierungsinformationen vom System gespeichert werden. Nach der Registrierung wird die Seite `welcome.wml` aufgerufen und der Benutzer kann die WAP-Dienste als angemeldeter User nutzen.
Die von Web nach WAP portierten Dienste sind Sammelmappen-, Verzeichnis- und Dokumenten-Verwaltung. Dabei können die im Web-Portal angelegten Dokumente im WAP-Portal zu den Sammelmappen über den Linkaufruf auf der Karte "Verzeichnisoptionen" hinzugefügt und aus den Sammelmappen über den Aufruf der Seite "Personalisieren" in `personalize.wml` entfernt werden. Es können außerdem auf Personalisieren-Seite neue Sammelmappen generiert und vorhandene umbenannt werden.
`directory.wml` stellt die Verzeichnisstruktur mit den sich darin befindlichen Dokumenten dar. Dabei kann durch die Verzeichnisbaumstruktur navigiert werden. Die in Verzeichnissen gespeicherten Dokumenten können abgerufen und dargestellt werden, sofern es sich um die WML- oder andere von den WAP-Geräten unterstützten Inhalte handelt.

Die Abbildung 25 enthält die WML-Seiten, die einen Teil der WAP-Portalerweiterung darstellen:

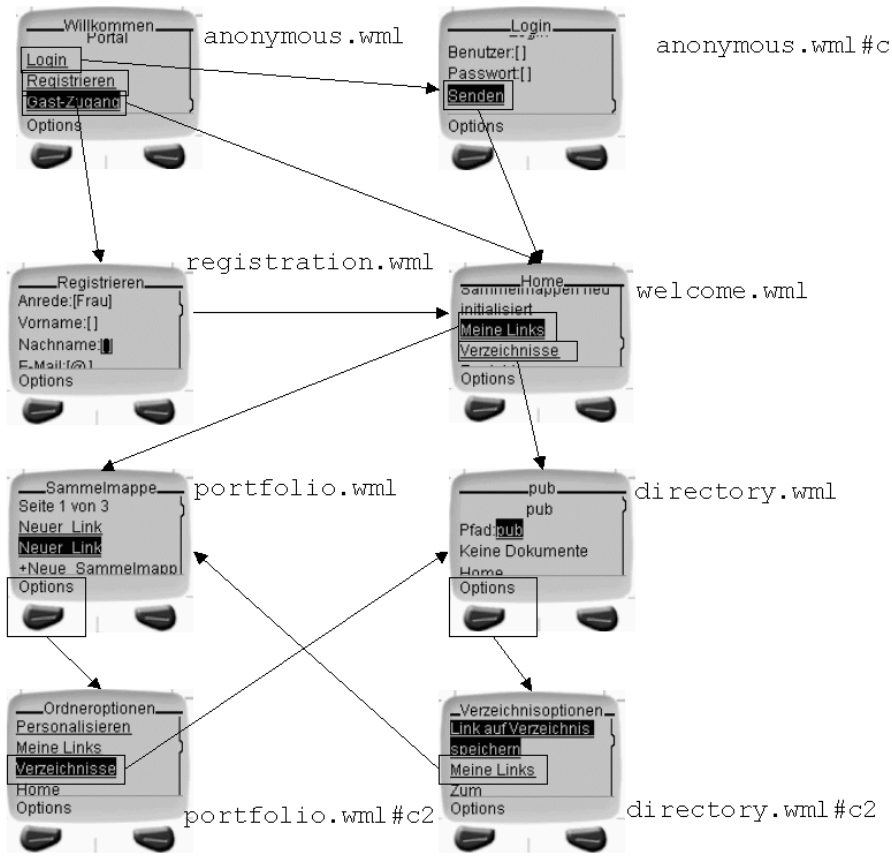


Abbildung 25: WAP-Portal-Funktionalität

Die Auswahl der nach WAP portierten Dienste unterliegt Kriterien und Beschränkungen eines mobilen Portals. D.h. es werden die Dienste des WAP-Portals portiert, die keine langwierigen und wiederholten Texteingaben bedürfen und deren Ausgaben kurz sind. Es wird also auf Textbearbeitungsfunktionen, Volltextbetrachtung, etc. verzichtet, da diese Funktionen auf einem mobilen Endgerät nicht komfortabel durchgeführt werden können. Es wird auch auf

Funktionen verzichtet, die keinen echten Nutzen für mobile Anwendungen, wie z.B. Personaldaten-Editieren, bringen oder einfach nicht möglich sind.
Die portierten Teile des Portals bilden eine Grundlage, auf der ein vollwertiges mobiles Portal mit zusätzlichen projektspezifischen Funktionen entwickelt werden kann.

5.2 Techniken zur Realisierung der Endgeräteunabhängikeit

Um die Geräeunabhängigkeit zu implementieren können drei verschiedene Ansätze verfolgt werden und zwar:

1. Entwicklung von generischen WML-Seiten
2. Entwicklung von für jedes einzelne Gerät optimierten Seiten
3. automatische Generierung von geräteoptimierten Seiten

Die folgenden Kapitel beschreiben die Vorgehensweise bei diesen Ansätzen und gehen auf die Vor- und Nachteile dieser Ansätze ein.
Es wird später auch deutlich, das die drei Ansätze parallel verwendet werden können, denn das Gesamtkonzept, die Speicherung der geräteoptimierten Dateiensätzen in einer Baumstruktur, erlaubt es, sowohl die WML-Seiten dynamisch zu generieren als auch einzelne Seiten manuell zu optimieren, wie es am Beispiel Nokia 7110 im Kapitel 5.3.3.5 deutlich wird. Der generische Ansatz wird auch verwendet. Es ist ein Dateiensatz in der Wurzel des logischen Baums vorhanden, der alle dem System unbekannten Geräte bedient. Dies ermöglicht das Bedienen auch solcher Geräte, für die es keine optimierten Seiten gibt.

5.2.1 Portable WAP-Anwendung

Die Entwicklung eines Dateien-Satzes, der generisch von allen WAP-fähigen Geräte verwendet wird, stellt softwaretechnisch die einfachste Lösung dar. Es sind prinzipiell keine zusätzlichen Massnahmen notwendig, da es durch die Trennung der Programmlogik von der Benutzeroberfläche (s. Kapitel 2.8.4) genügt, die WML-Templates zu entwickeln, ohne am Quellcode der Handler-Klassen Änderungen vornehmen zu müssen.
Es müssen jedoch Zusatzmechanismen implementiert werden, die z.B. Session-Tracking unterstützen, da der Session-Tracking-Mechanismus (s. Kapitel 5.3.3.1) sich geringfügig von dem des Web-Portals unterscheidet. Auch das Cache-Verhalten verschiedener Geräten muss durch entsprechende Mechanismen berücksichtigt werden.
Vom Entwicklungs- und Speicherungsaufwand ist dieser Ansatz am günstigsten, d.h. falls ein Dienst aus M Dateien besteht und N Geräte bedient werden müssen, so werden nur M Dateien entwickelt und gespeichert. Der Pflegeaufwand hat die Komplexität O(M), da maximal M Dateien bei einer Änderung modifiziert werden müssen.

5.2.2 Geräteoptimierte Seiten

Falls Geräte optimal bedient werden sollen, muss für jedes Gerätemodell ein eigener Dateiensatz, das für das jeweilige Modell optimiert ist, entwickelt werden. Dazu werden die Erkenntnisse, die im Kapitel 4 beschrieben werden, verwendet.
Ein weiterer Schritt zum automatischen Generieren von gerätespezifischen Templates ist, die Unterschiede im WML-Code zwischen den für verschiedene Endgeräte optimierten Dateien festzustellen. Dabei sollen evtl. die Endgeräte mit wenigen Unterschieden zu den Klassen

zusammengefasst werden. So haben die beiden Nokia-Geräte fast identischen WML-Code mit Ausnahme der Seiten, die das Setzen der Variablen und ein Bild enthalten, wie z.B. `anonymous.wml`-Seite (s. Kapitel 5.3.3.5). Auch Sony- und Palm-Browser können gemeinsam dieselben WML-Seiten benutzen, ohne dass die Benutzbarkeit eingeschränkt wird. Die Klassenbildung kann sinnvoll zum Bedienen von neuen Endgeräten sein, für die es noch keinen eigenen Dateien-Satz gibt, da die Geräte eines Herstellers meist auf gleicher Software basieren und ähnliches Verhalten aufweisen.
Um die richtigen WML-Seiten an das Gerät auszuliefern, muss das Gerät erkannt werden. Es wird anhand des `accept`-Headers festgestellt, der vom Client an den Server übermittelt wird und alle von ihm unterstützten MIME-Types enthält, ob es sich um einen WAP-fähigen Client handelt. Zum Festellen des genauen Gerätemodells wird der `useragent`-Header abgefragt und ausgewertet. Leider ist das Format des `useragent`-Header nicht genau spezifiziert, so dass man kein allgemeingültiges Verfahren zum Auffinden des Herstellernamens, des Models und der Version verwenden kann, um z.B. ein neues Gerät einer möglichst passenden Geräteklasse zuzuordnen. Man kann jedoch mit Hilfe eines Mustwers die bekannten Herstellernamen und Modellbezeichnungen erkennen und die Geräte den Geräteklassen zuordnen (s. Kapitel 5.4.2).
Um bei der Speicherung Redundanz zu vermeiden, wird vom Server eine Fall-Back-Lösung realisiert. Auf die Implementierung dieses Mechanismus wird im Kapitel 5.4.5 eingegangen.
Die Unterschiede im gerätespezifischen WML-Quellcode werden durch eine Gegenüberüberstellung deutlich. Das Beispiel zeigt den Unterschied zwischen zwei WML-Dateien, die an Nokia- und Siemens-Geräte angepasst sind, wobei die Bereiche mit einer Übereinstimmung des Quellcodes durch den weißen Hintergrund und solche, die den für das jeweilige Gerät optimierten Quellcode enthalten, durch den grauen Hintergrund hervorgehoben sind.

Siemens	Nokia
`<wml>`	`<wml>`
`<head>`	`<head>`
`<meta forua="true" content="no-cache" http-equiv="Cache-control"/>`	`<meta forua="true" content="no-cache" http-equiv="Cache-control"/>`
`</head>`	`</head>`
`<card id="c1" title="Portaloptionen">`	`<card id="c1" title="Portaloptionen">`
`<p align="center">`	
`<b>Portaloptionen</b>`	
`</p>`	
`<p>`	`<p>`
`<select>`	
`<option onpick="#c2">Landessprache</option>`	` `
	`<a href="#c2">Landessprache</a>`
`<option onpick="/de/myPortal/welcome.wml?s=$$(s)&c=$counter$">Home</option>`	` `
	`<a href="/de/myPortal/welcome.wml?s=$$(s)&c=$counter$">Home</a>`
`$[anonymous$`	`$[anonymous$`
`<option onpick="/de/myPortal/registration.wml?s=$$(s)&c=$counter$">Registrieren</option>`	` `
	`<a href="/de/myPortal/registration.wml?s=$$(s)&c=$counter$">Registrieren</a>`
`$]anonymous[$`	`$]anonymous[$`
`<option onpick="/de/myPortal/logoff.wml?s=$$(s)&c=$counter$">Abmelden</option>`	` `
	`<a href="/de/myPortal/logoff.wml?s=$$(s)&c=$counter$">Abmelden</a>`
`$anonymous]$`	`$anonymous]$`
`<option onpick="/de/help/info.wml?s=$$(s)&c=$counter$">Info</option>`	` `
	`<a href="/de/help/info.wml?s=$$(s)&c=$counter$">Info</a>`
`</select>`	
` `	` `
`</p>`	`</p>`

<do label="Zurück" type="accept"> <prev/> </do> </card> <card id="c2" title="Landessprache"> <onevent type="onenterforward"> <refresh> <setvar name="l" value="de"/> </refresh> </onevent> Landessprache Bitte wählen Sie: <select name="l"> <option value="de">deutsch</option> </select> Senden <do label="Zurück" type="accept"> <prev/> </do> </card> </wml>	<do type="prev"> <prev/> </do> </card> <card id="c2" title="Landessprache"> <onevent type="onenterforward"> <refresh> <setvar name="l" value="de"/> </refresh> </onevent> Bitte wählen Sie: <select name="l"> <option value="de">deutsch</option> </select> Senden <do type="prev"> <prev/> </do> </card> </wml>

Tabelle 15: Gegenüberstellung der Siemens- und Nokia-spezifischen WML-Templates

Bei der Gegenüberstellung der für verschiedene Geräte optimierten WML-Templates wird es deutlich, dass dieselben Funktionen mit Hilfe von verschiedenen Tags implementiert werden.
Um dem Anwendungsentwickler einen Leitfaden zu geben, welche Elemente und wie er diese in WML-Templates verwenden darf, werden Richtlinien spezifiziert. Diese Richtlinien schreiben unter anderem vor, welche Elemente und Attribute für welche Endgeräte verwendet werden und wie Session-Tracking realisiert wird. Eine Beschreibung der Richlinien wird im Kapitel 5.4.9 vorgenommen.
Der Entwicklungsaufwand ist im Vergleich zum generischen Ansatz sehr hoch: Bei einem aus M Dateien bestehenden Dienst und N Geräten, für die jedes der M Dateien optimiert wird, entsteht der Aufwand O(M*N). Genauso groß ist der Speicherungs- und Pflegeaufwand.

5.2.3 Automatische Generierung

Nach dem die Geräte zu Klassen zusammengefasst und die signifikanten Unterschiede im Quellcode festgestellt worden sind, soll eine Strategie zur automatischen Generierung von WML-Seiten entwickelt werden.
Der infoAsset Broker unterstützt `include`-Substitution, d.h. man kann die Platzhalter, die eine URL (`$#URL$`) enthalten, durch WML-Code-Fragmente, die in einer externen Datei vorliegen, ersetzen. Diese Datei kann auch alle vom Broker unterstützten Substitutionsarten (Kapitel 2.8.2) enthalten. Diese Möglichkeit ermöglicht die Code-Fragmente, die bei vielen Seiten gleich sind, in eine externe Datei auszulagern. Dadurch wird der Pflegeaufwand reduziert, da die `include`-Datei nur einmal modifiziert werden muss.
Eine weitere weitaus flexiblere Möglichkeit, die geräteoptimierten WML-Seiten zu generieren, ist XSLT. Da WML eine XML-Applikation ist, liegt die Wahl dieser Methode nahe. Es muss eine geräteneutrale WML-Seitenbeschreibung spezifiziert werden, die mittels XSL und XSLT in die gerätespezifische Form transformiert wird. Dabei wird der

Pflegeaufwand auf die Anzahl der geräteneutralen Seiten und die Anzahl der vorhandenen Geräteklassen reduziert, also O(M+N) anstelle von O(M*N) beim manuellen Anpassen.
In diesem Fall werden die Richtlinien dazu benutzt, die Entwicklung von geräteneutralen WML-Seiten zu beschreiben. Außerdem enthält der Style Guide die Informationen über relevante Geräteeigenschaften, Klassifikation, Problemfälle und Identifikationsmöglichkeiten der Endgeräte gegenüber dem Server.
Als geräteneutrales Format wird eine modifizierte Form von WML, das s.g. WMLPLUS-Format, definiert. WMLPLUS beschreibt die Seitenstruktur und enthält logische Elemente, die in WML mit verschiedenen Elementen realisiert werden können.

5.3 Richtlinien für Aufbau von WML- und WMLPLUS-Seiten

Für die Entwicklung von Diensten wird ein Style Guide verwendet. Er beschreibt, wie die geräteneutralen WMLPLUS-Seiten und gerätespezifischen WML-Dateien aufgebaut werden müssen. Für die Erstellung vom Style Guide werden die im Kapitel 4 gewonnenen Erkenntnisse über die WML-Implementierung verschiedener Geräte verwendet.

Der Style Guide beschreibt den Aufbau der WML- und WMLPLUS-Templates in einer Form, die für die Entwicklung von Diensten verwendet werden kann, ohne die Interna des infoAsset Broker zu kennen.

Der infoAsset Broker unterstützt zwei Methoden zum Erstellen von WML-Templates:

- Manuelle Optimierung
- Automatische Generierung der WML-Templates aus WMLPLUS-Dateien

Die manuelle Optimierung wird eingesetzt, falls es eine ad-hoc Optimierung wie im Falle der `anonymous.wml`-Datei von Nokia 7110 benötigt wird.
Die automatische Generierung mittels XSLT wird für die Generierung von vollständigen Dateiensätzen, die für ein Gerät optimiert sind, eingesetzt.
Der Unterschied zwischen den beiden Methoden besteht auch darin, dass die manuell optimierten Dateien nicht einer Geräteklasse zugeordnet werden dürfen, deren Dateien automatisch generiert werden, da sonst diese bei einer Neugenerierung überschrieben werden und die vorher gemachte Optimierung nichtig gemacht wird. Es muss für manuelle Optimierung eine neue Geräteklasse abgeleitet werden, die in der `properties.txt`-Datei eingetragen wird.

5.3.1 Aufbau der WML- und WMLPLUS-Seiten

WML-Templates entsprechen den WML-Spezifikationen mit Ausnahme der Verwendung von infoAsset Broker-Platzhaltern, die Dollarzeichen enhalten. Die WML-Templates können somit nicht direkt von einem WML-Browser geladen weden. Außerdem schreibt das Style Guide vor, wie die WML-Elemente verwendet werden und welche Elementenkombinationen nicht erlaubt sind.
Das WMLPLUS-Format lehnt sich sehr stark an das WML-Format an, hat aber eine etwas abgewandelte Struktur, wie z.B. das Fehlen der DTD im Dokumentenprolog. Einige zusätzliche WMLPLUS-Elemente erlauben das geräteunabhängige Behandeln von Konstrukten, die in WML auf geräteoptimierte Weise realisiert werden können.

Die Tabelle 16 bis Tabelle 28 enthalten Strukturen, die für jeweilige Geräte oder Formate unterschiedlich definiert sind. So kann eine Tabelle eine WML-Definition einer Struktur enthalten, falls diese Struktur für alle Geräte gleich ist, oder verschiedene Definitionen, für verschieden Gerätegruppen, falls es da Unterschiede gibt. Es gibt auch Fälle, bei denen das WML-Format mit dem WMLPLUS-Format übereinstimmt, dann ist nur eine Definition der Struktur vorhanden.

Die WML- und WNLPLUS-Templates haben folgende Struktur:

WML-Endgeräte	<?xml version="1.0" encoding="iso-8859-1"?> <!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN' 'http://www.wapforum.org/DTD/wml_1.1.xml'> <wml> <head> <meta http-equiv="Cache-Control" content="no-cache" forua="true"/> </head> *Kartendefinitionen* </wml>
WMLPLUS	<?xml version="1.0" encoding="iso-8859-1"?> <wmlplus> *Kartendefinitionen* </wmlplus>

Tabelle 16: Struktur der WML- und WMLPLUS-Templates

Die Kartendefinition beinhaltet die Dienstoberfläche, die die Benutzerschnittstelle bildet. Sie enthält im Falle von WML-Templates die WML-Elemente und infoAsset-Platzhalter und im Falle der WMLPLUS-Templates zusätzliche WMLPLUS-Elemente, die bei der automatischen Generierung mittels XSLT in WML-Elemente umgewandelt werden.

5.3.2 Kartendefinitionen

Eine Kartendefinition kann folgende Elemente enthalten:

- `card`-Elemente
- `do`-Elemente
- `template`-Tag

Bei den WML-Templates werden zwei Fälle unterschieden:

- Die Geräte mit Titel-Unterstützung
- Die Geräte ohne Titel-Unterstützung

Die Tabelle 17 zeigt die Definitionen der Karten, dabei stimmen die Definitionen für Nokia Geräte und im WMLPLUS-Format überein, da in beiden Fällen der Titel nicht explizit im Text definiert wird, wie das bei sonstigen Endgeräten der Fall ist.

Nokia und WMLPLUS	<card id="x" title="titelx"> Karteninhalt DoElemente </card>
Sonstige Endgeräte	<card id="x" title="titelx"> titelx Karteninhalt DoElemente </card>

Tabelle 17: WML-Template-Kartendefinition

Auf der WMLPLUS-Ebene wird der Titel nicht im Text angegeben, da bei der Generierung das Verhalten der Endgeräte in Bezug auf Titel-Unterstützung berücksichtigt wird.

Der Wert des `id`-Attributes `x` wird, wie folgt definiert:
`c` gefolgt von der Kartennummer wie z.B. `c1`, `c2`, ..., `cn`, wobei n die Anzahl der Karten in einem Deck ist. Dies ermöglicht eine möglichst ressourcenschonende und eindeutige Identifizierung und Referenzierung der Karten innerhalb eines Decks.
Die Zusammensetzung von `Karteninhalt` und `DoElemente` wird in Kapiteln 5.3.3 und 5.3.4 beschrieben.

5.3.3 Karteninhalt

5.3.3.1 Sitzungsverwaltung

Für die Sitzungsverwaltung wird eine clientseitige Variable verwendet, die bei dem Aufruf der ersten Seite vom Server initialisiert wird.

WML-Endgeräte und WMLPLUS	`<onevent type="onenterforward"><refresh>`
	`<setvar name="s" value="$sessionid$">`
	`</refresh>`
	`</onevent>`

Tabelle 18: Sitzungsverwaltung: Setzen der Session-Id

Dieser Code-Fragment wird als erster direkt nach dem öffnenden `card`-Tag platziert.

Alle Links müssen einen `s=$$(s)`String enthalten, der die Session-Id mittels des URL-Parameters an den Server übergibt.

WML-Endgeräte und WMLPLUS	`<a href="`*`ZielUrl`*`?s=$$(s)">`

Tabelle 19: Sitzungsverwaltung: Übergabe der Session-Id an den Server

Dabei wird die WML-Variable `$s` als `$$s` in einem infoAsset Broker WML- oder WMLPLUS-Template verwendet (`$$` wird bei der Substitution durch `$` ersetzt).
Die Links, die eine Karte innerhalb eines Decks oder außerhalb des eigenen Servers aufgerufen werden, bedürfen des Parameters `s` nicht, da keine Kommunikation mit dem infoAsset Broker stattfindet.

5.3.3.2 Links

Der infoAsset Broker unterscheidet zwischen den Einzelllinks und den logisch zusammengefassten Links, den Linklisten. Die Linklisten werden verwendet, wenn eine Seite, wie z.B. `anonymous.wml`, eine konstante Konstellation von Links enthält. Einzellinks werden dann verwendet, wenn die Anzahl der Links in einer Liste zur Entwicklungszeit nicht bekannt ist, d.h. wenn die Links dynamisch verwaltet werden, wie das der Fall bei den Sammelmappen und Verzeichnissen ist.
Die Einzelllinks werden wie folgt definiert:

WML-Endgeräte	Karteninhalt1 Linktext Karteninhalt2
WMLPLUS	Karteninhalt1 Linktext Karteninhalt2

Tabelle 20: Definition von Einzellinks

Die Pfadangabe erfolgt relativ zum Wurzelverzeichnis und der Parameter `s` ist obligatorisch mit Ausnahme der Fälle, die im Kapitel 5.3.3.1 erläutert wurden.

Die Definition von Linklisten ist folgende:

Siemens	<select> <option onpick="*ZielURL1*">Titel 1</option> <option onpick="*ZielURL2*">Titel 2</option> ... </select>
Alle sonstigen Endgeräte	 *Titel 1* *Titel 2* ...
WMLPLUS	<linklist> <link href="*ZielURL1*">*Titel 1*</link> <link href="*ZielURL2*">*Titel 2*</link> </linklist>

Tabelle 21: Linklistendefinition

Linklisten erlauben bei Siemens-Geräten die Aktivierung der Links mittels der Zifferntasten. Die entsprechenden Ziffern werden vor den Links angezeigt.

5.3.3.3 Texteingabe

Texteingabefelder dienen dazu, die Benutzereingaben an den Server zu übermitteln. Die URL-Parameternamen sind serverseitig definiert und sollten möglichst kurz gewählt werden., Ebenso müssen die WML-Variablennamen kurz gewählt werden, um die Bytegröße der WML-Seiten möglichst gering zu halten. Die `input`-Tags müssen sich innerhalb eines eigenen `p`-Elementes befinden, da es sonst Probleme bei Siemens-Geräten gibt, falls ein Link vor einem `input`-Tag definiert ist.

WML-Endgeräte und WMLPLUS	 <input type=type1 name=name1 value=value1 size=size1 format=format1/> <input type=type2 name=name2 value=value2 size=size2 format=format2/> Senden

Tabelle 22: Eingabefeld-Definition

Falls der Wert eines Eingabefeldes servergesteuert vorbelegt werden soll, wird es mit Hilfe der serverseitigen Substitution initialisiert, indem man `value="$substitution$"` setzt.

Das `format`-Attribut soll verwendet werden, obwohl es mangelhaft unterstützt wird, da es zumindest das Umschalten zwischen Buchstaben und Ziffern sichert. Dabei ist das Sony-Gerät eine Ausnahme, es ignoriert das `format`-Attribut.

5.3.3.4 Auswahllisten

Die Auswahllisten dienen dazu, aus einer vordefinierten Liste einen oder mehrere Einträge auszuwählen. Die Auswahllisten werden aufgrund derselben Probleme wie Texteingabefelder in eigenem `p`-Element definiert.

WML-Endgeräte und WMLPLUS	`<p>`
	`<select name=name1 value=value1>`
	`<option value=choice1>Titel 1</option>`
	`<option value= choice2>Titel 2</option>`
	`</select>`
	`<select name=name2 value=value2 multiple="true">`
	`<option value= choice1>Titel 1</option>`
	`<option value= choice2>Titel 2</option>`
	`</select>`
	`</p>`

Tabelle 23: Definition der Auswahllisten

Soll eine Mehrfachauswahlliste definiert werden, wird der Wert des `multiple`-Attributes auf `true` gesetzt. Andernfalls wird das `multiple`-Attribute ausgelassen.

5.3.3.5 Bilder

Bilder werden WML-konform verwendet. Es soll immer ein alternativer Text vorhanden sein, falls z.B. die Bildanzeige vom Benutzer ausgeschaltet wird.

WML-Endgeräte	`<img src="/de/skin/images/document1.wbmp" alt="text"/>`

Tabelle 24: Einbetten eines Bildes

Will man das gleiche Erscheinen der Bilder auf allen Endgeräten erzielen, so werden zusätzliche `br`-Tags eingefügt. In diesem Fall ignorieren die Nokia-Geräte die überflüssigen `br`-Tags und erzeugen keine Leerzeilen.

WML-Endgeräte	`Zeile 1` ` <img src="/de/skin/images/document1.wbmp" alt="text"/>` ` Zeile 3`

Tabelle 25: Uniforme Darstellung der Bilder auf verschiedenen Geräten

Nokia 7110 darf die Elemente zum Setzen einer Variablen und das Anzeigen eines Bildes auf der gleichen Karte nicht enthalten (s Kapitel 5.3.3.5), wie das folgende Beispiel zeigt:

```
<card id="c1" title="Willkommen">
<onevent type="onenterforward">
<refresh>
<setvar name="s" value="$sessionId$"/>
</refresh>
</onevent>
<p align="center">Willkommen im
```

```
<br/><img src="/de/skin/images/smapco.wbmp" alt="smapCo"/>
<br/>...
```

Deshalb wird eine neue Geräteklasse initialisiert, die alle Dateien von Nokia-Geräteklasse erbt, außer den Dateien, die `setvar`- und `img`-Tags enthalten. Diese werden durch manuell optimierte Dateien ersetzt:

```
<card id="c1" title="Willkommen">
<onevent type="onenterforward">
<refresh>
<setvar name="s" value="$sessionId$"/>
</refresh>
</onevent>
<p align="center">Willkommen im
<br/>smapCo
<br/>...
```

Die Initialisierung einer neuen Geräteklasse wird im Kapitel 5.4.4 behandelt und im Kapitel 5.4.5 wird es auf das Beispiel des Nokia 7110 eingegangen.

5.3.4 DoElemente

Mit `do`-Elementen werden die Sofkeys initialisiert, in diesem Kapitel wird gezeigt, wie die verschiedenen Sofkey-Arten bei verschiedenen Endgeräten und im WMLPLUS-Format initialisiert werden.

5.3.4.1 Rücksprung zur zuletzt angezeigten Karte

Jede Karte sollte eine Möglichkeit zum Rücksprung zur vorherigen Karte haben. Dies wird mittels eines `prev`-Elements erreicht. Diesbezüglich werden zwei Gerätegruppen unterschieden, bei denen die beiden Rücksprungmöglichkeiten mit `do`-Elementen vom verschiedenen Typ realisiert werden. Die Tabelle zeigt, wie die Rücksprungfunktionalität definiert wird.

Nokia	`<do type="prev"><prev/></do>`
Alle anderen Endgeräte	`<do type="accept" label="Zurück"><prev/></do>`
WMLPLUS	`<do type="prev">Ignoriert</do>`

Tabelle 26: Rücksprung-Implementierung

5.3.4.2 "Option"-Softkey (Kartenorientierte Navigationsunterstützung)

Ein weiteres `do`-element vom Typ `options` enthält die Referenz auf eine Karte, die eine Reihe von weiterführenden Links, also ein Navigationsmenü, enthält. Es sind typischerweise Standardoptionen, z.B. Home, Landessprache, Abmelden, Registrieren, etc. Wird ein Navigationsmenü mehrfach verwendet, so kann es auf ein separates Deck ausgelagert werden. Im Falle, dass nur ein Teil des Navigationsmenüs gleich bleibt und ein anderer von der referenzierenden Karte abhängt, also kontextabhängig ist, können die include-Platzhalter eingesetzt werden, um einen WML-Fragment, der in einer Textdatei vorliegt, in eine WML-Seite textuell einzubinden. Die `include`-Datei kann ihrerseits alle vorhandenen Substitutionsarten enthalten.
Die Tabelle zeigt die Definition eines `option`-Softkeys:

Nokia	`<do type="options" label="+Option"><go href="ZielURL"/></do>`
Alle sonstigen Endgeräte und WMLPLUS	`<do type="options" label="Option"><go href="ZielURL"/></do>`

Tabelle 27: Implementierung des "Option"-Softkeys

Das Plus-Zeichen bei Nokia-Geräten dient dazu, den Eintrag optisch von den geräteeigenen Einträgen hervorzuheben.
Aufgrund der Begrenzung der Sofkey-Anzahl bei den meisten Geräten auf zwei, wird auf die Unterstützung von weiteren `type`-Attributen verzichtet. Sind weitere Navigationseinträge nötig, so können diese mit Hilfe von Links realisiert werden.

5.3.5 template-Tag

Falls ein `do`-Element auf allen Karten eines Decks identisch sein soll, d.h. Typ, Label und Task sind gleich, so kann dieses Element auch in einem Template definiert werden. Der `template`-Tag wird vor der ersten Karte des Blocks `Kartendefinitionen` definiert.
Es soll dabei berücksichtigt werden, dass bei den Palm- und Sony-Geräten die Reihenfolge der im `template`-Tag vorhandenen Softkeys unter Umständen nicht die gewünschte ist. Die Template-Softkeys haben eine niedrigere Priorität als die in der Karte definierten Sofkeys und erscheinen je nach Endgerät meistens als letzte.

WML-Endgeräte und WMLPLUS	`<template>`
	`DoElemente`
	`</template>`
	`Kartendefinitionen`

Tabelle 28: Verwendung des `template`-Tags

5.4 Softwaretechnische Umsetzung

Um die Geräteunabhängigkeit zu implementieren, ist eine Erweiterung der Klassen in verschiedenen Schichten des infoAsset Brokers notwendig:

- Ggf. geringfügige Anpassung der Handler
- Geräteerkennung, Geräte- und Gerteklassenverwaltung, unter anderem die Realisierung der logischen Baumstruktur und des Fallback-Algorithmus, Suchstrategie (`UserAgent`, `UserAgentClass`, `UserAgentFactory`).
- `properties.txt`-File
- Abweichendes WML-Sitzungsverwaltungsmechanismus
- Deckgrößenbeschränkung
- Transformation: XSLT, XSL und Neugenerierung der WML-Templates im Web-Portal
- Mechanismen, die abnormales Cache-Verhalten kompensieren (Zähler, serverseitiges Verwalten der History)

Die Abbildung 26 zeigt die Übersicht über die Komponenten, die die Geräteunabhängigkeit realisieren. Es sind zwei entkoppelte Teile erkennbar:

Der Teil rechts beschreibt die Generierung der Templates. Der Teil links enthält Komponennten, die für die Verwaltung der generierten Templates, Erkennung des User Agents und für das Zurückliefern der richtigen Templates zuständig.

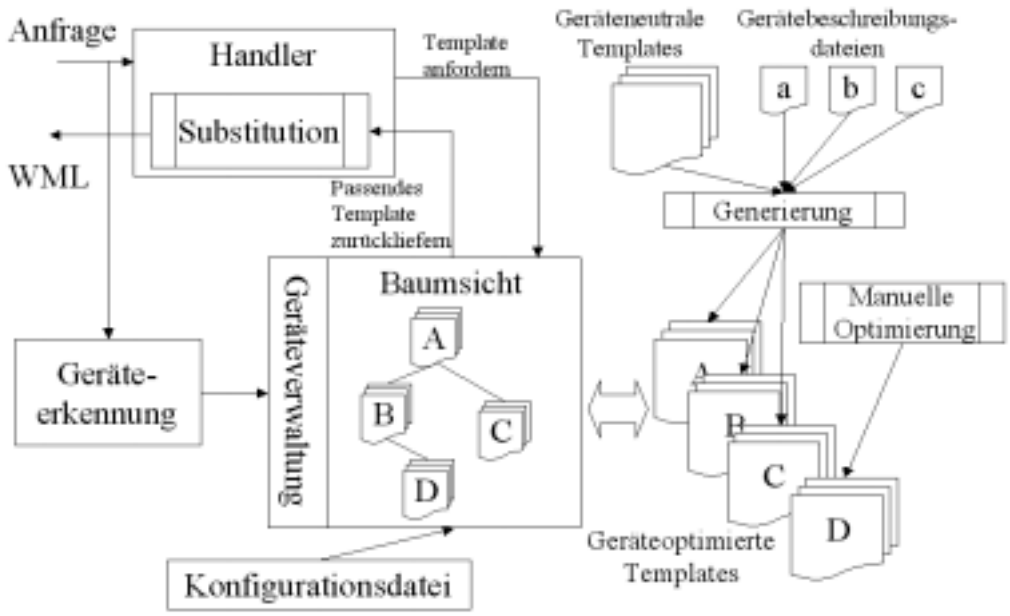


Abbildung 26: Überblick über die Realisierung

Die einzelnen Komponenten dieser Abbildung werden in folgenden Kapiteln im Detail beschreiben.

5.4.1 Endgeräteklassifikation

Die Endgeräte werden zu Klassen zusammengefasst. Ein Endgerät wird einer bestehenden Klasse zugeordnet, falls es optimal den WML-Code, der für diese Geräteklasse generiert wurde, darstellen kann. Kann ein Gerät nicht einer bestehenden Geräteklasse zugeordnet werden, wird eine neue Klasse angelegt. Es ist jedoch wichtig, möglichst wenige Geräteklassen anzustreben, um den Entwicklungsaufwand gering zu halten. So werden bei einigen Geräten Kompromisse eingegangen, um trotz der suboptimalen Lösung in Bezug auf Darstellung und Benutzbarkeit die Geräte zu einer Klasse zuzuordnen. Hier ist eine Übersicht über die Geräteklassen und denen zugeordneten Geräte.

Endgeräteklasse	**Name der Klasse im infoAsset Broker 1.1**	**Getestet mit folgenden Endgeräten der Endgeräteklasse**
HMTL-Endgeräte	HTML	Internet Explorer
WML-Endgeräte	WML	Unbekannte WAP-fähige Geräte
Siemens-Endgeräte	Siemens	Siemens C35i, Siemens S35i, Siemens M35i
Nokia-Endgeräte	Nokia	Nokia 6210, Nokia 7110, Yospace.com HTML Emulator
Sonstige Endgeräte	Other	Sony CMD-Z5, Windows Up-Browser Emulator mit Motorola Timeport Skin
Palm-Endgeräte	Palm	Palm Pilot 100 mit AUSystem-Browser.

Tabelle 29: Geräte und Klassenzuordnung

So erlaubt der Verzicht auf die Anpassung der Darstellung auf Palm-spezifische Eigenschaften wie die Größe des Displays, die Zuordnung der Palm-Endgeräte zu der Klasse Other und Reduzierung der Anzahl von Endgeräteklassen.
Die Geräte werden mit Hilfe einer hierarchischen Baumstruktur klassifiziert:

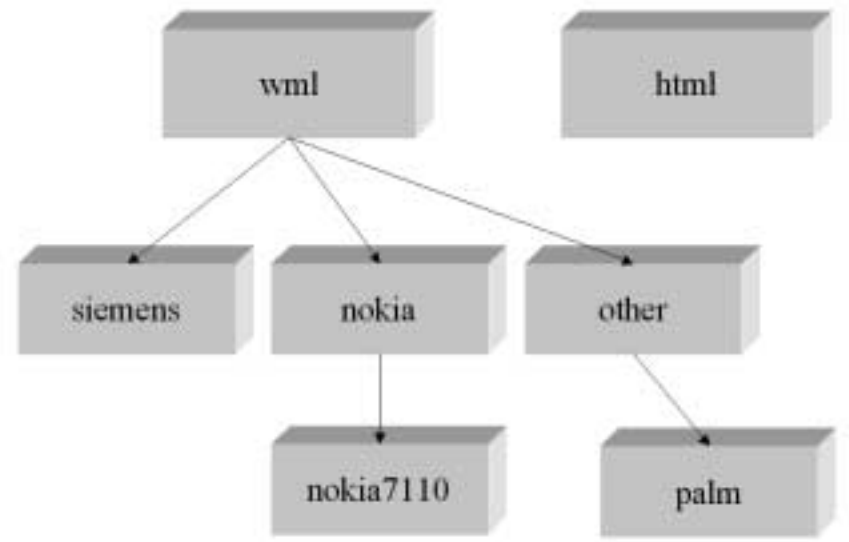


Abbildung 27: Anordnung der Geräte in einem logischen Baum

Dies ermöglicht die Einordnung der neuen Geräte an die Stelle im logischen Baum, deren umgebende Knoten die größte Ähnlichkeit des angepassten Quellcodes aufweisen. Für das Endgerät heißt es, dass nur die abweichenden Seiten implementiert werden müssen. Ein Beispiel dafür sind das Gerät Nokia 7110 und die Geräteklasse Nokia, deren WML-Seiten weitgehend identisch sind, außer dass die `anonymous.wml`-Datei des Nokia 7110-Gerätes sich durch die Verwendung von Bildern (s. Kapitel 4.3.9) unterscheidet und als einzige Seite speziell für Nokia 7110 implementiert wird. Alle anderen Seiten werden von der Nokia-Geräteklasse vererbt.
Die logische Baumstruktur wird in der `properties.txt`-Datei gespeichert, deren Aufbau im Kapitel 5.4.4 beschreiben wird.

5.4.2 Automatische Identifikation der Geräte

Die Identifikation der Geräte erfolgt mit Hilfe des `UserAgent`-Strings, der vom Client an den Server im `UserAgentString`-Header gesendet wird und serverseitig ausgewertet werden kann. Die einzelnen Geräte haben folgende User Agent-Strings:

Endgerät	**User Agent-String**
Siemens C35i	`SIE-C3I/3.0 UP/4.1.16m` *`Gatewaystring`*
Siemens S35i	`SIE-S35/1.0 UP/4.1.8c` *`Gatewaystring`*
Nokia 6210	`Nokia6210/1.0 (`*`xx.yy`*`)`
Nokia 7110	`Nokia7110/1.0 (`*`xx.yy`*`)`
Yospace.com HTML Emulator	`Nokia 7110 v1.3 (compatible; YOSPACE SmartPhone Emulator 1.2)`
Sony CMD-Z5	`Mozilla/1.22 (compatible; MMEF20; Sony CMD-Z5)` *`Gatewaystring`*
Palm Pilot 100 mit	`AUR PALM WAPPER (WAP 1.1)` *`Gatewaystring`*

AUSystem-Browser	
Windows Up-Browser Emulator mit Motorola Timeport Skin	`MOT-CB/0.0.18 UP/4.0.10 UP.Browser/4.0.10-XXXX UP.Link/4.1.HTTP-DIRECT`
Kein Browser: UP.Link-Gateway	*`Gatewaystring`* `= UP.Link/4.1.0.6`

Tabelle 30: User Agent-Strings

Dabei sind die User Agent-Strings verschiedener Geräte nicht standardisiert, d.h. man kann bei einem unbekannten Gerät nicht ohne weiteres die Informationen wie den Herstellernamen, die Software, die Software- und Hardware-Version sicher extrahieren.
Außerdem wird bei Verwendung eines UP.Link-Browsers in Kombination mit einem UP.Browser-basierten Endgerät und einigen anderen, wie AU-System-Browser, ein vom WAP-Gateway stammender String angehängt (s. untere Zeile in der Tabelle 30).
Das Gateway hat einen zusätzlichen Einfluss auf die maximale zulässige Deckgröße.
Die Deckgrößenbeschränkung wird im Kapitel 5.4.9 behandelt.

Um einem Endgerät geräteoptimierte WML-Seiten anbieten zu können, muss das erkannte Gerät dem Server bekannt sein. Durch den Vergleich des `AuserAgent`-Strings mit den aus dem `properties.txt`-File eingelesenen `pattern`-Attribute im `UserAgent`-String versucht der Server das Endgerät zu erkennen. Schlägt der Versuch fehl, weil z.B. ein neues nicht in der Konfigurationsdatei enthaltenes Gerät benutzt wird, versucht der Server anhand des `Accept`-Headers festzustellen, ob der Client WAP-fähig ist oder ob es sich um einen HTML-Client handelt.

5.4.3 Endgeräteverwaltung

Die Schnittstellen und Klassen, die für die Client-Erkennung und Verwaltung zuständig sind, sind die `UserAgent`-Schnitstelle, `UserAgentClass`-Schnittstelle und `UserAgentFactory`-Klasse.
`UserAgentFactory`-Klasse liest die `useragentclass`-, `useragent`- und `restrictions`-Einträge aus dem `properties.txt`-File (s Kapitel 5.4.4) und ist dafür zuständig eine hierarchische Struktur aus den Klassen, die die Schnittstellen `UserAgentClass` und `UserAgent` implementieren, aufzubauen. Sie stellt Funktionen zur Verfügung, die einen User-Agent-Objekt anhand der vorhandenen Merkmalen, wie `UserAgent`-String, der Endung des angeforderten Dateinamens oder `Accept`-String, zurückliefern. Das Klassendiagram in der Abbildung 28 veranschaulicht die Zuständigen Klassen und Schnittstellen:

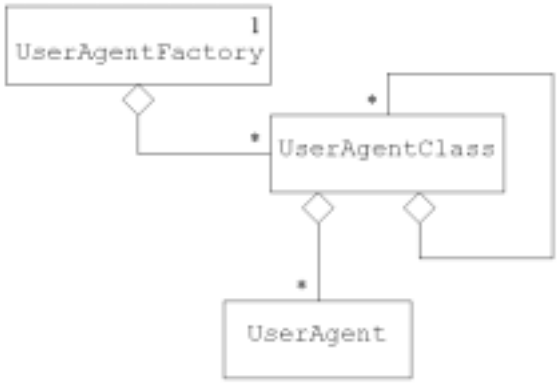


Abbildung 28: Für die Endgeräteverwaltung zuständigen Klassen bzw. Schnittstellen

Die Klasse `UserAgentFactory` enthält `get-`, `getUserAgent` und `getUserAgentClasses`-Funktionen, mit deren Hilfe alle in der Konfigurationsdatei enthaltenen Geräte und Geräteklassen zurückgeliefert werden. Die `get`-Funktion sucht in der hierarchischen Struktur nach einem User Agent mit der vorgegebenen Id und liefert eine `UserAgent`-Instanz zurück. Die `getUserAgent`-Funktion liefert ein `UserAgent`-Objekt, das am besten mit den übergebenen Parametern übereinstimmt. Die `GetUserAgentClasses`-Funktion liefert alle dem System bekannten Geräteklasse zurück.
Die Schnittstelle `UserAgentClass` bzw. die Klassen, die diese Schnittstelle implementieren, stellt Funktionen bereit, die die in der `properties.txt`-Datei definierten Eigenschaften der Geräteklasse zurückliefern:

- `getClassId` liefert die Id der Geräteklasse wie `html`, `wml`, `nokia`, `siemens` zurück.
- `getClassName` liefert den Name der Geräteklasse wie `Nokia WAP Device` zurück.
- `getParent` liefert die übergeordnete Geräteklasse zurück.
- `getPreferredTextExtension` liefert die Dateinamenserweiterung der Textformate wie `.html`, `.wml`, `.txt` zurück.
- `performsCaching` liefert einen Wert vom Typ `boolean` zurück, der das Verhalten des Endgerätes in bezug auf Caching widerspiegelt.

Die `UserAgent`-Schnitstelle beschreibt den Client und enhält folgende Funktionen, die dessen Eigenschaften zurückliefern:

- `getId` liefert die server-interne Id des Clients.
- `getMaxByteSize` liefert die maximale Größe des Decks, die verarbeitet werden kann.
- `getName` gibt den in `properties.txt`-Datei definierten Namen zurück.
- `getUserAgentClass` liefert die übergeordnete Geräteklasse.

5.4.4 Konfiguration

Die Konfigurationsdatei enthält eine Reihe von Informationen, die die Anpassung des infoAsset Brokers an die verschiedenen Umgebungen ermöglicht.
Für die Geräteunabhängigkeit ist die Konfiguration der Clients ein wichtiger Teil. Die Konfiguration beschreibt die Eigenschaften der Clients, deren Geräteklassenzugehörigkeit und damit auch deren Anordnung im logischen Baum, der die Beziehung der Geräte zueinander und die WML-Code-Ähnlichkeit darstellt. Hier ist ein entsprechender Auszug aus der Konfigurationsdatei:

```
useragentclass.html.name = HTML Browser
useragentclass.html.extension = htm,html,js,css,gif,jpg,jpeg
useragentclass.html.performsCaching = true

useragentclass.wml.name= WAP Device
useragentclass.wml.extension = wml,wbmp
useragentclass.wml.performsCaching = false

useragentclass.nokia.parent= wml
useragentclass.nokia.name= Nokia WAP Device
useragentclass.nokia.performsCaching = false

useragentclass.nokia7110.parent= nokia
useragentclass.nokia7110.name= Nokia 7110 WAP Device
useragentclass.nokia7110.performsCaching = false
```

Der obere Ausschnitt beschreibt die Endgeräteklassen und ordnet sie in eine logische Baumstruktur ein. Dabei wird die Verzeichnisstruktur, in der die HTML- und WML-Templates liegen, auf die Baumstruktur abgebildet, wie die Abbildung 27 im Kapitel 5.4.1 verdeutlicht.

Dabei liegen die Verzeichnisse `html`, `wml`, `siemens`, `nokia`, `nokia7110`, `plam`, `other` in demselben Unterverzeichnis in einer Ebene, d.h. die logische Baumstruktur wird nicht durch die Verzeichnisstruktur, sondern durch die `parent`-Werte in der `properties.txt`-Datei widergespiegelt.
Das `performscaching`-Attribut gibt an, ob das Endgerät das im Kapitel 5.4.8 beschriebene Verhalten aufweist. D.h. falls dieses Attribut den Wert `"false"` hat, wird der Mechanismus aktiviert, der wiederholtes Ausführen einer Aktion verhindert.
Der nächste Abschnitt der `properties.txt`-Datei enthält die Informationen über die Geräte:

- den Namen der Geräteklasse, der das Gerät angehört,
- den Namen des Gerätes, unter dem es dem System bekannt ist,
- den Suchstring, der zur Identifizierung des Gerätes anhand des User-Agent-Strings herangezogen wird, und
- untrstützte maximale Deckgröße.

```
useragent.siemensc35i.class = siemens
useragent.siemensc35i.name = Siemens c35i
useragent.siemensc35i.pattern = SIE-C3I
useragent.siemensc35i.maxbytesize = 1506

useragent.nokia6210.class = nokia
useragent.nokia6210.name = Nokia 6210
useragent.nokia6210.pattern = Nokia6210
useragent.nokia6210.maxbytesize = 1372

useragent.nokia7110.class = nokia7110
useragent.nokia7110.name = Nokia 7110
useragent.nokia7110.pattern = Nokia7110
useragent.nokia7110.maxbytesize = 1372
```

Diesem Gerät werden alle zur Entwicklungszeit unbekannten WAP-fähigen Geräte zugeordnet.

```
useragent.wml.class = wml
useragent.wml.name = Unknown WML Browser
useragent.wml.pattern = --this will never match--
useragent.wml.maxbytesize = 1300
```

Das Gerät repräsentiert HTML-Browser.

```
useragent.html.class = html
useragent.html.name = HTML Browser
useragent.html.pattern = Mozilla
```

Um den Einfluss des WAP-Gateways bzgl. der Deckgrößenbeschränkung zu berücksichtigen, wird eine Zeichenkette, mit deren Hilfe das WAP-Gateway erkannt wird, und eine maximale Deckgröße vorgegeben.

```
restrictions.uplink41gateway.pattern = UP.Link/4.1
restrictions.uplink41gateway.maxbytesize = 1461
```

Die Informationen werden für die Erkennung, Verwaltung der Engeräte und Generierung von geräteoptimierten WML-Seiten verwendet.

5.4.5 Redundanzfreie Speicherung und Suchstrategie

Durch die Anordnung der Geräteklassen in einer hierarchischen Baumstruktur, die Verwandtschaftsbeziehungen der Geräte und insbesondere die Ähnlichkeit des WML-Codes zum Ausdruck bringt, ist es möglich, nur die Seiten zu spezifizieren, die auch geräteoptimiert werden müssen. Die identischen Seiten werden in den Eltern-Knoten gespeichert, wobei der Spezialisierungsgrad zu den Blättern hin steigt.
Um die am besten angepasste Datei zu finden, ist es erforderlich, den logischen Baum von den Blättern aus zur Wurzel hin zu durchsuchen, z.B. im Falle von Nokia 7110 werden laut der Konfigurationsdatei folgende Unterverzeichnisse nacheinander durchsucht, bis die angeforderte Datei gefunden wird:

- das `nokia7110`-Unterverzeichnis, das nur die an Nokia 7110 angepasste Datei `anonymous.wml` enthält,
- das `nokia`-Unterverzeichnis, das den vollen Satz von WML-Templates enthält, die für Nokia-Geräte optimiert sind,
- das `wml`-Unterverzeichnis, das den vollen Satz von WML-Templates enthält, die generisch sind. Sie werden aufgerufen, wenn z.B. der für Nokia optimierter Datensatz nicht vollständig ist, weil eine Optimierung nicht nötig ist oder keine optimierten Dateien, z.B. im Falle eines neuen Gerätes, generiert wurden.

Der logische Baum, der das repräsentiert, sieht folgendermaßen aus:

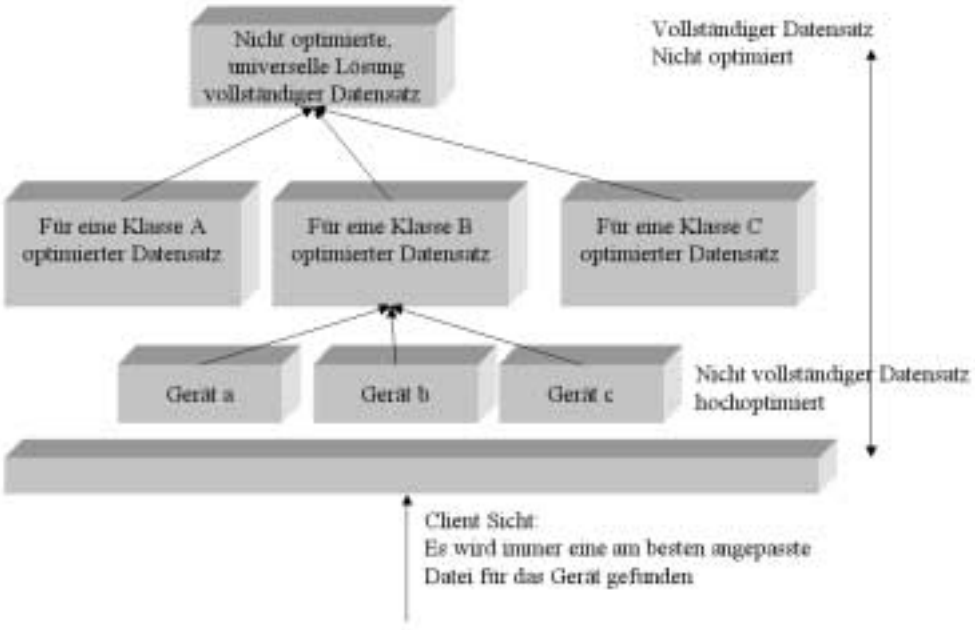

Nicht optimierte, universelle Lösung vollständiger Datensatz
Vollständiger Datensatz Nicht optimiert
Für eine Klasse A optimierter Datensatz
Für eine Klasse B optimierter Datensatz
Für eine Klasse C optimierter Datensatz
Gerät a
Gerät b
Gerät c
Nicht vollständiger Datensatz hochoptimiert
Client Sicht:
Es wird immer eine am besten angepasste Datei für das Gerät gefunden

Abbildung 29: Logischer Baum - Speicherungs- und Suchstrategie

Diese Vorgehensweise kann man auch auf Mehrsprachigkeit eines Projektes und auf verschiedene Projekte, die von einem Basisprojekt abgeleitet werden, anwenden. Dadurch können Redundanzen vermieden werden und beim Verwenden von verschiedenen Sprachen wird immer auf eine voreingestellte Standardsprache ausgewichen, falls die Seiten in der angeforderten Sprache nicht vorhanden sind.

Die Abbildung 30 zeigt wie eine Projekt-, Sprachen- und Geräteverwaltung in einer Baumstruktur organisiert werden kann:

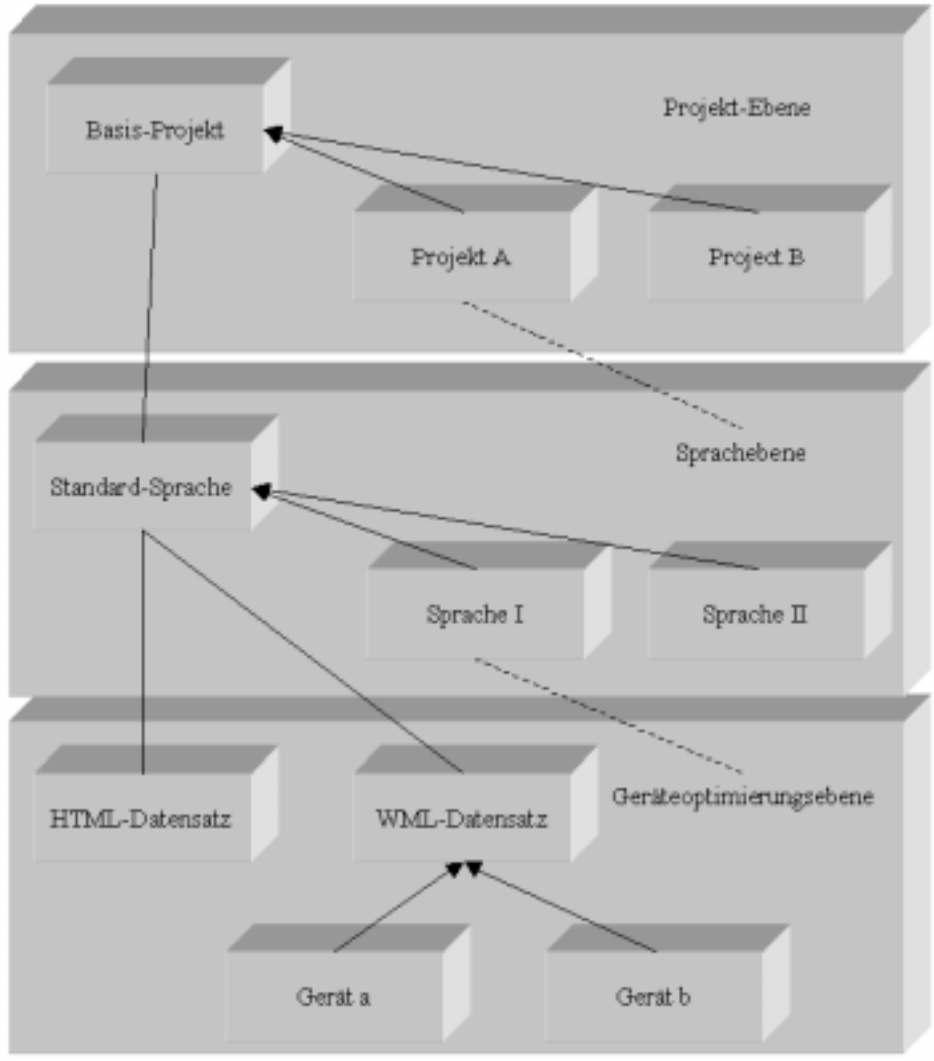


Abbildung 30: Projekt, Mehrsprachigkeit, Medien- und Geräteunabhängigkeit

5.4.6 Generierung von Templates

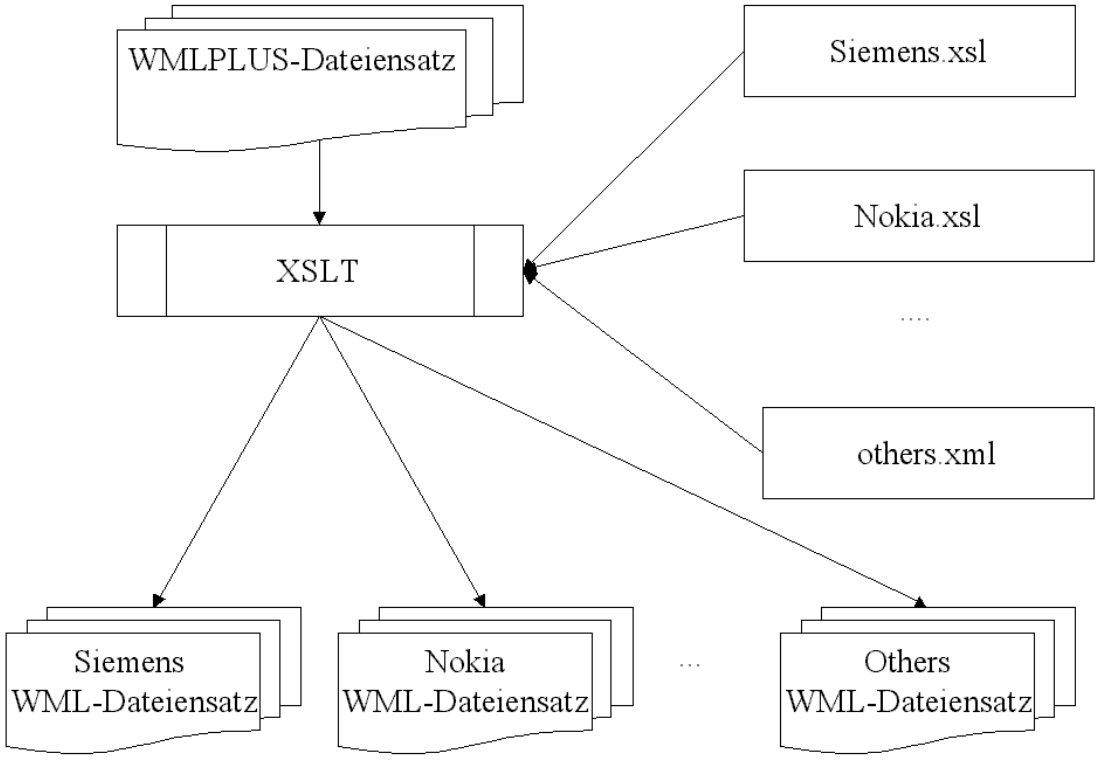
Für die Generierung von WML-Templates wird XSLT verwendet. Als XSLT-Prozessor wird Xalan-Implementierung [Kay2000] eingesetzt. Die WML-Templates können vom Administrator im Web-Portal neu generiert werden.

Xalan ist über die `XSLTProzessor`-Klasse im `util`-Package eingebunden und enthält Methoden, die Transformation einer einzelnen XML-Datei erlauben. Die Generierung der Datensätze übernimmt die `GenerateWMLHandler`-Klasse. Dieser Handler kann im Web-Portal von einem autorisierten Benutzer aufgerufen werden, um aus den WMLPLUS-Templates die geräteoptimierten WML-Templates automatisch zu erstellen.

Die Abbildung 31 beschreibt, welche Dateien und Dateiformate für die Generierung von gerätespezifischen Dateien notwendig sind.

Eine WMLPLUS-Datei enthält alle in WML üblichen Elemente und ist durch eine Reihe von spezifischen Elementen ergänzt, wie `linklist` und `link`, um den Anforderungen, die für die Geräteunabhängigkeit notwendig sind, gerecht zu werden. Außerdem weicht die Verwendung von do-Elementen vom Typ `prev` in WMLPLUS von WML ab.
Auf eine DTD wird dabei bewusst verzichtet, damit die Erweiterung mit neuen Elementen einfach ist. Es müssen jedoch trotzdem bei der Verwendung des WMLPLUS-Formates Regeln beachtet werden, damit bei der Konvertierung mittels XSLT Dateien im gültigen WML-Format generiert werden können. Auf die Restriktionen, die das WML-Format betreffen, und auf das WMLPLUS-Format wird im Kapitel 5.3 eingegangen.

Abbildung 31: Generierung von WML-Templates mittels XSLT



5.4.7 Cache-Verhalten und Zähler

Der infoAsset Broker verwaltet einen zusätzlichen Parameter in der Session-Schicht, der bei jedem neuen Aufruf einer Seite inkrementiert und als URL-Parameter in jeden Link eingesetzt wird. Der Zähler hat zwei Aufgaben:

- Er verhindert, dass die Seiten aus dem Cache geladen werden. Durch die Inkrementierung des Zählers werden die gleichen URL-Referenzen vom WML-Browser als neue URLs behandelt und somit bei jedem Aufruf aktualisiert.
- Er wird zum Detektieren der Rückwärtsnavigation verwendet, da bei der Rückwärtsnavigation die URL aus dem History-Speicher des Clients verwendet wird und den alten Zählerstand hat. Man kann durch den Vergleich des serverinternen Wertes mit dem Wert aus dem Anfrage-String feststellen, ob die Anfrage aus dem History-Speicher des Clients kommt oder mittels eines Links erfolgt. Der Sachverhalt ist ist im Kapitel 5.4.8 verdeutlicht.

Der Zähler wird bei dem ersten Aufruf der Startseite vom Server neu initialisiert und bei jedem weiteren Aufruf einer Seite inkrementiert und mittels Substitution als Parameter in jede URL dieser Seite eingesetzt.

5.4.8 Detektieren der Rückwärtsnavigation nach dem Aufruf einer Seite mit Seiteneffekt

Alle Endgeräte verfügen über einen Seiten-Cache, in dem WML-Seiten gespeichert werden, und eine URL-History, die die aufgerufene URL-Einträge speichert. Dies ermöglicht das clientseitige Speichern von WML-Seiten und einen schnellen Zugriff auf diese Seiten, ohne dabei die Anfrage an den Server zu senden. Doch einige Geräte, z.B. Nokia 7110 und 6210, unterstützen zwar den clientseitigen Cache, jedoch wird die Seite bei Rückwärtsnavigation vom Server angefordert. Dies hat Auswirkungen, falls die WML-Seite, die vorher aufgerufen wurde, seiteneffektbehaftet ist. Die Seite wird nochmals aufgerufen und die Aktion wird wiederholt. Die Auswirkungen reichen vom wiederholten Anlegen eines Verzeichnisses bis zu einem Versuch des Entfernen eines bereits gelöschten Verzeichnisses.
Deswegen wird vom infoAsset Broker ein Mechanismus implementiert, der erkennt, in welche Richtung die Navigation stattfindet und ob es sich um einen Aufruf einer Seite mit Seiteneffekt handelt.
Dazu wird eine serverseitige Speicherung der URL-Strings aller aufgerufenen Seiten mit Seiteneffekt implementiert. Um die Rückwärtsnavigation zu detektieren, wird der Wert der Zähler-Variablen serverseitig mit dem Wert des Zähler-Parameters aus dem URL-String verglichen. Im Falle der Rückwärtsnavigation ist der Wert des Zähler-Parameters kleiner als der Wert der serverseitigen Zähler-Variablen.
Die Abbildungen veranschaulichen die Rückwärtsnavigation und die damit verbundenen Probleme:

- Rückwärtsnavigation mit Pufferung von Inhalten (Abbildung 32),
- ohne Pufferung von Inhalten (Abbildung 33) und
- ohne Pufferung von Inhalten (Abbildung 34), aber mit Detektierung von Rückwärtsnavigation.

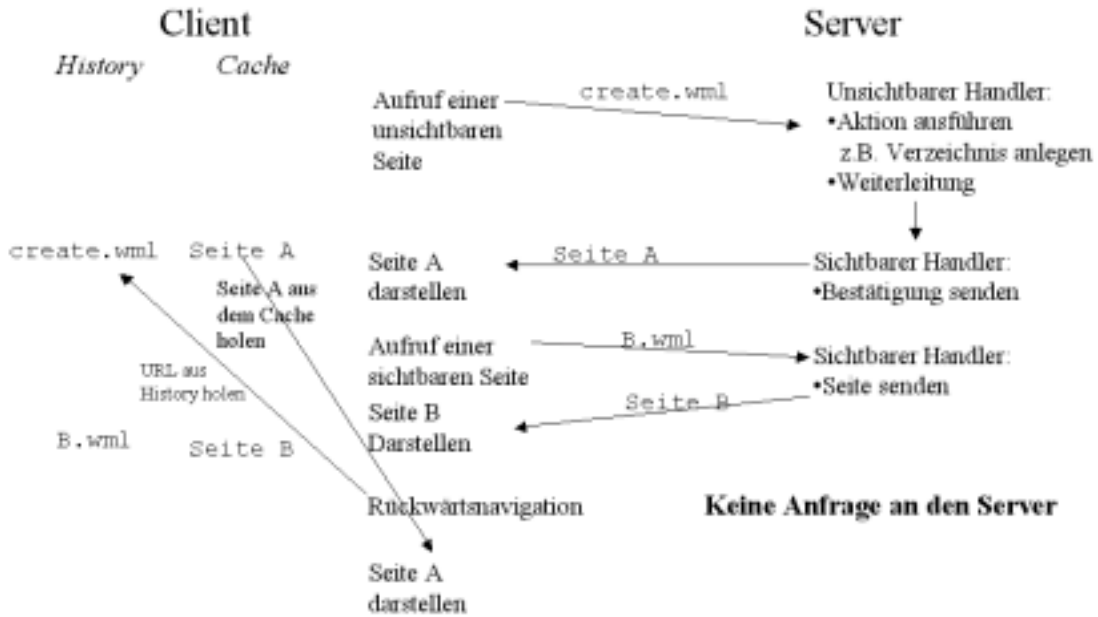


Abbildung 32: Rückwärtsnavigation mit Pufferung der Seiten

Die Abbildung 32 zeigt das Verhalten eines Clients, der sowohl die URLs als auch die Seiten puffert und die Inhalte bei der Rückwärtsnavigation aus dem Speicher lädt. Dabei sind keine zusätzlichen serverseitigen Mechanismen notwendig, da kein Aufruf von Handlern mit Seiteneffekt bei der Rückwärtsnavigation stattfindet.

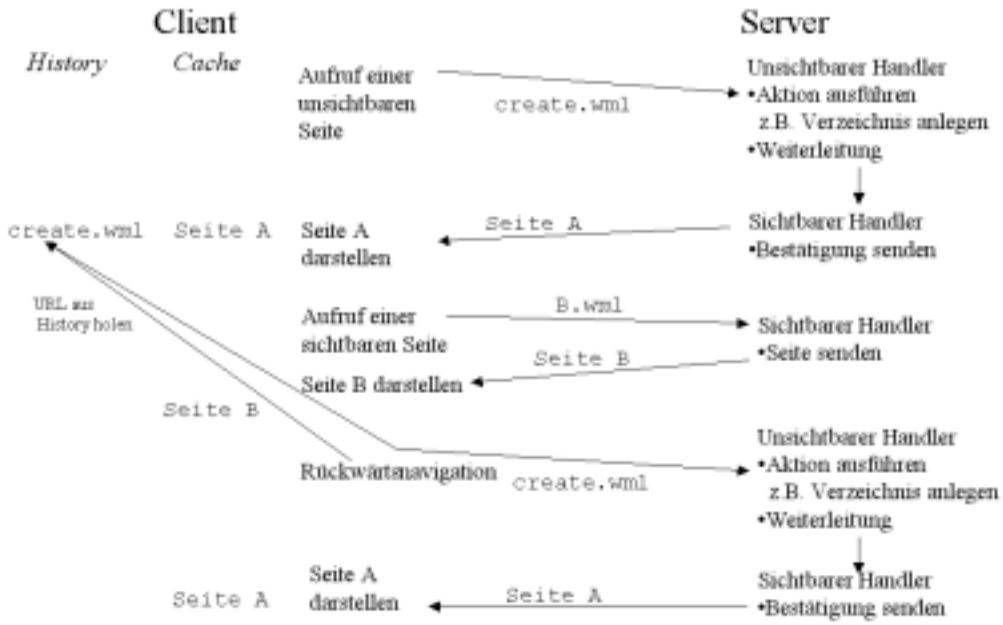


Abbildung 33: Rückwärtsnavigation ohne Pufferung von Seiten

Die Abbildung 33 zeigt den Fall, bei dem die Pufferung von URLs stattfindet. Die Inhalte werden jedoch vom Server angefordert. Dies hat zur Folge, das eine alte URL wiederholt einen Handler mit Seiteneffekt aufruft, mit der Konsequenz, dass die Aktion noch mal ausgeführt wird.

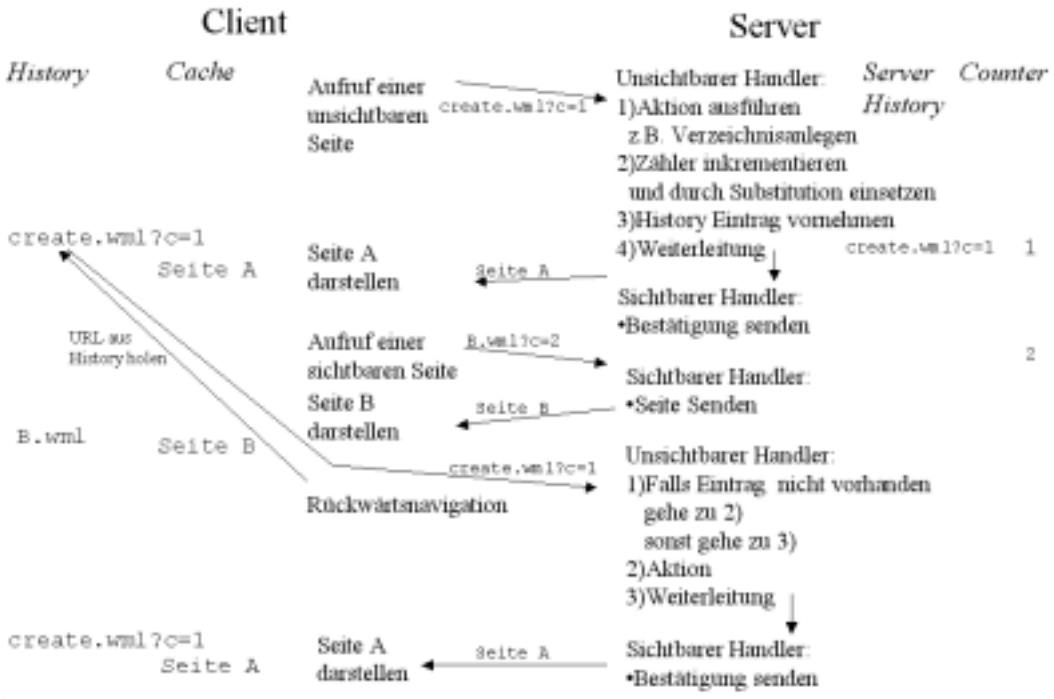


Abbildung 34: Serverseitige Mechanismen zur Detektierung der Rückwärtsnavigation

Die Abbildung 34 zeigt die Implementierung der Rückwärtsnavigation-Detektierung. Es wird serverseitig eine History geführt, in der die URLs der aufgerufenen Seiten mit Seiteneffekt gespeichert werden. Dadurch, dass die URLs einen Counter-HTTP-Parameter enthalten, kann man durch den Vergleich des aktuellen Zählerstandes mit dem in der URL enthaltenen feststellen, ob es sich um eine URL aus der History des Clients oder um eine aus dem direkten Linkaufruf handelt.
Um die Geräte, die den Seitencache erwartungsgemäß verwalten, von den Geräten, die eine gepufferte Seite jedoch vom Server anfordern, zu unterscheiden, wird in der Konfigurationsdatei `properties.txt` ein entsprechender Eintrag gemacht (s. Kapitel 5.4.4).

5.4.9 Deckgrößenbeschränkung und Seitenlistenvorlagen

Alle verwendeten Geräte weisen eine Deckgrößenbeschränkung auf. Das Attribut im `properties.txt`-File enthält für jedes Gerät den Wert `maxbytesize`, der nicht überschritten werden darf. Die Deckgrößenbeschränkung bringt mit sich ein Problem bei den Seiten, die dynamisch generiert werden. Vor allem der Einsatz von Listen mit einer großen Anzahl von Einträgen kann dazu führen, dass die maximale Deckgröße überschritten wird und von den Geräten nicht verarbeitet werden kann. Für diese Zwecke wird die Substitution, die mit Seitenlistenvorlagen arbeitet, eingesetzt. Dabei wird die zu substituierende Liste so auf mehrere Listen aufgeteilt, dass die Einzellisten die maximale Deckgröße nicht übersteigen. Die Einzellisten werden auf mehrere Seiten aufgeteilt, zwischen denen navigiert werden kann. In der Seitenlistenvorlage werden folgende Platzhalter definiert:

- `$currentPage$` wird durch die Nummer der aktuell angezeigten Seite ersetzt (beginnend mit 1).
- `$numberOfPages$` wird durch die Anzahl der insgesamt angezeigten Seiten ersetzt (`numberOfPages` > 0).
- `$[isFirst$` ist eine bedingte Substitution, die nur für die Seite 1 ausgeführt wird.
- `$[isLast]$` ist eine bedingte Substitution, die nur für die letzte Seite ausgeführt wird.
- `$previousPage$` wird durch die Nummer der Vorgängerseite ersetzt ("", falls `isFirst`).
- `$nextPage$` wird durch die Nummer der Folgeseite ersetzt ("", falls `isLast`).
- `$numberOfElements$` wird durch die Anzahl der Elemente (Treffer) substituiert.
- `$[isEmpty$` ist eine bedingte Substitution, die nur für eine leere Anzahl von Elementen ausgeführt wird.

Eine Seitenlistenvorlage kann nicht in einer anderen Vorlage (`ConditionalTemplate`, `ListTemplate`, `PageListTemplate`) geschachtelt werden.
Eine Seitenlistenvorlage darf beliebig viele geschachtelte Vorlagen enthalten (`ConditionalTemplate`, `ListTemplate`).
Beim Aufruf einer Seitenlistenvorlage über einen HTTP-Request wird ein optionaler Parameter `idx=Seite` übergeben, der die Nummer der Seite angibt, die anzuzeigen ist. Fehlt der Parameter `idx`, wird die erste Seite angezeigt.

5.5 Generalisierung von Geräteunabhängigkeit

Die Aspekte, die bei der Implementierung der Geräteunabhängigkeit eingeflossen sind, wie Generierung vom geräteoptimierten Quellcode und Verwaltung von Geräteklassen, können auf Medienunabhängigkeit verallgemeinert werden.
Es muss ein medienunabhängiges Format definiert werden, mit dem Dienste allgemein beschrieben werden können. Die Generierung von medienspezifischen Seiten kann mittels XSLT erfolgen, wobei die XSL-Dateien Informationen über die Medienformate enthalten.
Die Schwierigkeit besteht darin, ein medienneutrales Format zu entwickeln, das die Dienste medienunabhängig beschreibt, aber auch gleichzeitig, die Eigenschaften des jeweiligen Standards nicht einschränkt.
Wenn man beispielhaft zwei Formate wie HTML und WML betrachtet, so stellt man fest, dass beide Mark-Up-Sprachen sind, die die gleichen Sprachstrukturen aufweisen.
Die wesentlichen Mark-Up-Elemente, die die Funktionalität eines Dienstes ausmachen stimmen im Wesentlichen überein. Doch in Bezug auf Layout und Design gibt es überhaupt keine Übereinstimmungen. Dies bedeutet, dass die Funktionen und das Aussehen des Dienstes separat behandelt werden sollen. So muss die Gestaltung der Oberfläche unabhängig von der Benutzerlogik durchgefürt werden und durch XSLT oder andere Mechanismen, wie z.B. Dream-Weaver-Templates, eingebunden werden.
Ein weiterer Aspekt ist der Einsatz von Skriptsprachen. Diese müssen auf jedem Fall medienspezifisch entwickelt werden, da die Clients sehr unterschiedlich sind und keine automatische Konvertierung bzw. Generierung zulassen.
Insgesammt erscheint das Problem der Medienunabhängigkeit komplexer als die Geräteunabhängigkeit, lässt sich aber durch die Erweiterung der bestehenden Konzepte realisieren.

6 Zusammenfassung, Bewertung & Ausblick

In dieser Arbeit wurde gezeigt, wie man bei einem Problem vorgeht, das so unscharfe Rahmenbedingungen mit sich bringt, wie einerseits der WAP-Standard mit WML-Elementen-Definitionen, die laut WML-Spezifikation von Geräteherstellern implementiert werden können aber nicht müssen, und andererseits die Gerätehersteller, die eine Minimallösung implementiert haben, so dass ein WAP-Anwendungsentwickler gar nicht absehen kann, ob die von ihm entwickelte Anwendung auch auf einem Endgerät funktioniert. Die Schwierigkeit bei der Geräteoptimierung liegt nicht nur in der unterschiedlichen Unterstützung von WML-Elementen, sondern auch darin, dass die meisten Informationen, ob und wie die WML-Elemente und deren Attribute unterstützt werden, nicht von den Geräteherstellern selbst stammen, sondern erst durch Tests ermittelt werden mussten. Die Gerätehersteller lieferten mit ihren eigenen Style Guides zwar eine gewisse Anregung, geben aber keinen Anhaltspunkt für die Implementierung der Geräteunabhängigkeit. Auch verschiedene Fehler in der Implementierung erforderten mehr serverseitige Verarbeitung.
Es bleibt zu wünschen, dass sowohl der Wortlaut der zukünftigen WAP-Spezifikationen als auch die Qualität der Implementierung sich zum Positiven entwickelt.
Auch genereller Umfang an WML-Elementen, vor allem Layout-Funktionen, und Multimedia-Unterstützung soll an die steigende Leistungsfähigkeit der modernen Mobilfunknetze angepasst werden, damit WAP zu HTML konkurrenzfähig wird.

Doch der wirkliche Vorteil der mobilen Dienste kann nur dann ausgereizt werden, wenn die Schnittstelle zu den Lokalisierungsdiensten spezifiziert und in WAP integriert ist und die damit zusammenhängenden datenschutzrechtlichen Fragen geklärt sind. Technisch gesehen gibt es schon seit langem solche Mechanismen, die das Lokalisieren eines Mobilfunkteilnehmers ermöglichen.
Unter dem Strich überzeugt der WAP-Standard durch seinen soliden Grundkonzept, der auf Wiederverwendung von vorhandenen Internetstrukuren und Optimierung der Übertragung über die Mobilfunkstrecke aufsetzt, zeigt aber Schwächen bei der WML-Spezifikation und deren Implementierung. Dies erschwert die Entwicklung von gerätetunabhängigen WAP-Anwendungen.

Die WAP-Erweiterung von infoAsset Broker zeigt einen möglichen Weg, die Geräteunabhängigkeit zu implementieren. Die vom Hause aus auf der multi-tier-Architektur basierte Software kann in den entsprechenden Schichten angepasst und ergänzt werden. Die Konzepte, die für die Geräteunabhängigkeit entwickelt wurden, können auch auf die Medienunabhängigkeit erweitert werden. So kann sowohl die Optimierung der bestehenden Dienste für neue Endgeräte als auch die Anpassung der Dienste an die neuen Formate wie HDML mit minimalem Entwicklungsaufwand durchgeführt werden.
Durch die fehlerhafte Implementierung des WAP-Standardes mussten einige Zusatzmechanismen vor allem für die Steuerung des Cache-Speichers implementiert weden.
Weitere Erweiterung wie User Agent-Verwaltung und Organisation der Dateiensätze in einer baumartigen Struktur können dafür eingestzt werden, um Medienunabhängigkeit aber auch Multilingualität und Multiprojektfähigkeit zu erzielen.
Auf der Basis des infoAsset Brokers wurde eine funktionsfähige WAP-Portalerweiterung entwickelt, die verschiedene WAP-fähigen Clients bedienen kann. Durch den Einsatz von XSLT ist es möglich, ohne großen Entwicklungsaufwand die Palette der unterstützten Geräte jede Zeit zu erweitern.

Der nächste Schritt bei der Entwicklung des infoAsset Brokers in Bezug auf Unterstützung von verschiedenen Clients kann die Entwicklung der automatischen Generierung von medienspezifischen Formate sein, wie in Kapitel 5.5 beschrieben.

Abkürzungen

ASP Active Server Pages
BXML Binary eXtendable Mark-up Language
CDF Cannel Definition Format
cHTML compact HTML
DTD Document Type Definition
EDGE Enhanced Data Service for GSM Evolution
GPRS General Packet Radio Service
GSM Global System for Mobile Communications
HSCD High Speed Circuit Switched Data
HTML Hypertext Mark-Up Language
HTTP Hypertext Transfer Protocol
JSP Java Server Pages
MIME Multipurpose Internet Mail Extension
MSC Mobile Switching Center
OS Operating System
OSI-ISO Open System Interconnection-International Standard Organisation
OTA Over The Air-Protocol
PAP Push Access Protocol
PDA Personal Digital Assistent
PI Push Initiator
PPG Push Proxy Gateway
TCP/IP Transmission Control Protocol/Internet Protocol
TLS-SSL Transport Layer Security-Secure Socket Layer
UMTS Universal Mobile Telecommunications System
URI Uniform Resource Indicator
URL Uniform Resource Locator
WAE Wireless Application Environment
WAP Wireless Application Protocol
WBMP Wirless Bitmap
WBXML WAP Binary XML
WDP Wireless Datagram Protocol
WML Wireless Mark-Up Language
WMLScript Wireless Mark-Up Language Script
WSP Wireless Session Protocol
WTA Wireless Telephony Application
WTAI Wireless Telephony Application Interface
WTLS Wireless Transport Layer Security
WTP Wireless Transaction Protocol
VCF Visit Card Format
XML eXtendable Mark-up Language
XSL extendable Style-Sheet Language
XSLT XSL Transformation

Glossar

Card oder Karte ist eine von WML benutzte Metapher. Eine Karte bildet eine logische Einheit von zusammenhängenden Elementen, wie Text, Eingabefelder oder andere Benutzerschnittstellenelemente.

CBS (Cell Broadcast System) erlaubt das Verschicken von Nachrichten an mehrere mobile Telefone, deren Benutzer sich an einem bestimmten Ort aufhalten.

COO (Cell of Origin) erlaubt, den Benutzer innerhalb einer GSM-Zelle zu lokalisieren. Die generierten Ortsangaben sind sehr ungenau, erlauben aber eine Reihe von Diensten, wie z.B. ortsabhängige Tarife oder auch die Angabe des Standortes, um standortabhängige Informationen mittels WAP anzubieten.

Deck oder Stapel ist eine WML-Datei, die eine oder mehrere Karten enthält.

Event, oder auch intrinsic event, ist ein von WML definierter Mechanismus, um auf die vom Benutzer oder vom User Agent generierte Aktionen reagieren zu können. Z.B beim Anspringen einer Karte, das Setzen von Variablen.

GPRS (General Packet Radio Service) stellt eine paketvermittelte Technik zur Verfügung, die Übertragungsraten von bis zu 115 KBit/s realisiert. GPRS kann problemlos in die bestehenden Netze integriert werden (MTR2000).

GSM (Global System for Mobile Communications) ist der Standard für digitale mobile Funknetze.

GPS (Global Positioning System) ist ein Sateliten-basiertes System, das sehr genaue Bestimmung der Position erlaubt.

HSCSD (High Speed Circuit Switched Data) erhöht die bestehende Übertragungsrate von 9,6 KBit/s auf bis zu 14,4 KBit/s pro Kanal durch eine neue Modulationsart. Ebenfalls ist eine Kanal-Bündelung theoretisch von bis zu acht Kanälen möglich [C4-2000].

Location Based Services sind solche Dienste, die die aus den COO-Angaben oder mit Hilfe von GPS gewonnenen Ortsangaben verwenden, um die ortsabhängigen Informationen anzubieten.

Origin-Server ist ein Web-Server, auf dem die WAP- oder Web-Anwendung gespeichert ist.

SMS (Short Message Service) erlaubt das Empfangen und Senden von maximal 160 alphanumerischen Zeichen pro Sendung (MTR2000).

Softkey ist ein Benutzerschnittstellenelement, dessen Funktion und Beschriftung frei programmiert werden kann.

Tasks sind die in WML definierten Elemente, die das Wählen eines Links, Aktualisieren des internen Zustandes des User Agents und Navigieren zur vorhergehenden Seite ermöglichen.

Template ist eine vom infoAsset Broker verwendete Datei, die eine vorgenerierte Datei im jeweiligen Format darstellt, z.B. HTML oder WML, und infoAsset Platzhalter enthält, die mit dynamischen Inhalten zur Laufzeit ersetzt werden können.

UMTS (Universal Mobile Telecommunications System) ist die 3. Generation des GSM-Standards und erlaubt eine breitbandige Mobilkommunikation bis zu 2 MBit/Sekunde [MTR2000].

User Agent (UA) ist ein Endgerät, der einen WAP-fähigen Browser implementiert. Ein UA kann auch ein Programm sein, das auf einem PC läuft, wie z.B. ein Emulator.

User Agent Pofile (UAProf) ist ein im WAP-Standard spezifiziertes Schema, das die Charakteristika der mobilen Geräte beschreibt.

WAP (Wireless Application Protocol) ist ein offener, herstellerübergreifender Standard für die Entwicklung drahtloser IP-Dienste und ermöglicht eine Kommunikation zwischen mobilen Kommunikationsgeräten (Mobiltelefon, PDA, Pager) und Internet-Anwendungen. WAP ist unabhängig von der zur Übertragung verwendeten Netztechnologie (GSM, GPRS, UMTS) und somit sehr gut skalierbar und zukunftssicher [MTR2000].

WAP-Gateway ist ein Server, der das Internet und das Funknetz verbindet. Das WAP-Gateway kann auch zusätzliche Funktionen übernehmen, wie z.B. das Umwandeln von HTML-Seiten in WML, Herausfiltern oder Konvertieren von Grafiken etc.

WBXML (WAP Binary XML) ist ein Format, der zur Übertragung von XML-Dokumenten über die Mobilfunkstrecke eingesetzt wird. Durch die Umwandlung wird die Dateigröße und dadurch die Wartezeit beim Download reduziert.

WML (Wireless Markup Language) ist eine im WAP-Standard spezifizierte Markup-Sprache, ähnlich HTML, die auf XML basiert. Sie wird zur Darstellung von Inhalten auf mobilen Geräten eingesetzt.

WMLScript ist eine Skript-Sprache, ähnlich JavaScript, mit deren Hilfe clientseitige Verarbeitung auf mobilen Geräten möglich ist.

Literatur- und Quellenverzeichnis

1. MTR2000
   Dr. Materna GmbH www.materna.de/technik.html

2. NOK99
   Service Developer's Guide for the Nokia 7110, Nokia, November 1999

3. C4-2000
   Chip März 2000 Artikel Mit dem Handy ins Internet R. Sablowsky

4. Har97
   Java Network Programming, Elliote Rusty Harold, O'Reilly, February 1997

5. Risch2000
   WAP und WML Wireless Web – Das neue Internet, Ray Rischpater, Galileo Press, Juni 2000

6. Booch99
   Das UML-Benutzerhandbuch, Grady Booch, Addison-Wesley, 1999

7. Chan99
   The Java Developers Almanac, Patric Chan, Addison-Wesley, 1999

8. JLW96
   Web Programmierung, Jmsa/Lalani/Weakley, Franzis', 1996

9. Klute
   Das World Wide Web, Rainer Klute, Addison-Wesley

10. Imm2000
    Das Große Buch WAP, Immler, Data Becker, 2000

11. Areh2000
    Porfessional WAP, Charls Arehart, WROX, 2000

12. SCNEd2000
    Mobile Networking with WAP, SCN Education B. V. (Eds), 2000

13. See2000
    XML, Michael Seeberger-Weichselbaum, bhw, 2000

14. XSLT99
    XSLT Transformation (XSLT) Version1.0, W3C, 1999

15. JavaXML20000
    Java PI for XML Processing, SUN, 2000

16. KAU-94
PC-Netze und Workgroup Computing, Franz-Joachim Kauffels, Markt&Technik Verlag, 1994

17. iMode2000
Mobile Solution, Tle Talk Sonderausgabe Nr. 3/2000 Artikel Mobil – Made in Japan (i-Mode)

18. c't2000
c't 2000, Heft 19, Turbolader für Funk-Bits, Dr. Dirk Nikolai, Klaus Daniel, Dr. Edgar Kühn, 2000

19. VCAL96
"vCalendar – The Electronic Business Calendaring and Scheduling Format", Version 1.0, The Internet Mail Consortium, September 18, 1996, http://www.imc.org/pdi/vcal-10.doc

20. VCARD96
"vCard – The Electronic Business Card", Version 2.1, The Internet Mail Consortium, September 18, 1996, http://www.imc.org/pdi/vcard-21.doc

21. WMLS2000
WAP-193-WMLScript Language Specification, Wireless Application Protocol Forum, Juni 2000

22. WML2000
WAP-191-WML Language Specification, Wireless Application Protocol Forum, Februar 2000

23. PUSH99
WAP Push Architectural Overview, Wireless Application Protocol Forum, November 1999

24. UAPROF
Wireless Application Group, User Agent Profile Specification, November 1999

25. RDF2000
Resource Description Framework (RDF) Schema Spezifikation, W3C Candidate Recommendation, März 2000

26. Phone2000
Application Style Guide: For GSM 900 and 1800, Phone.com, Juni 2000

27. CLIPP
Web-Clipping-Architekture, www.palm.com

28. Kay2000
XSLT Programmer's Reference, Michael Kay, Wrox, 2000